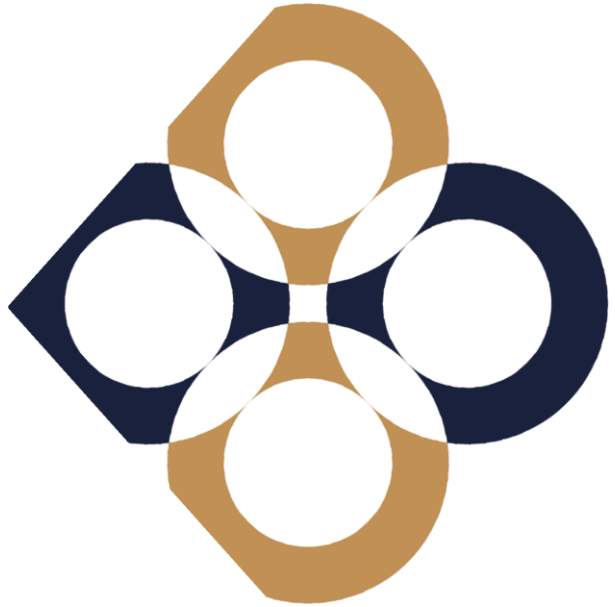
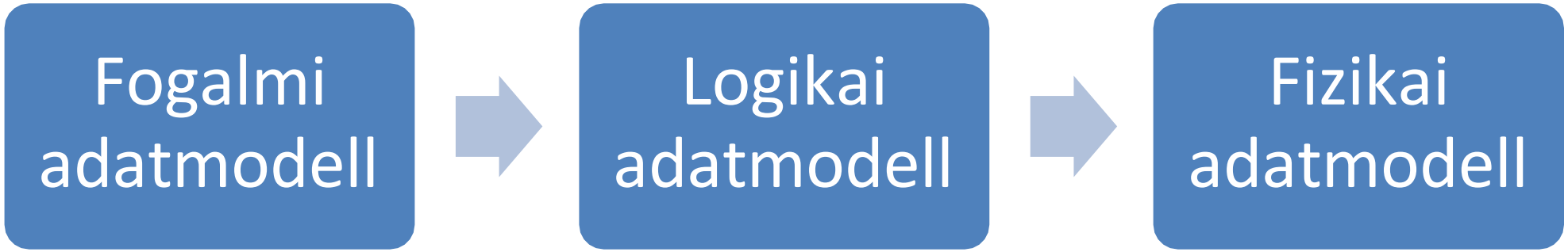
BUDAPESTI CORVINUS EGYETEM

Adatbázisok előadás 04

Fizikai adatbázisterv
Adatbázisok fejlesztése



A tervezés lépései



Egyed – kapcsolat modell (ER-modell)

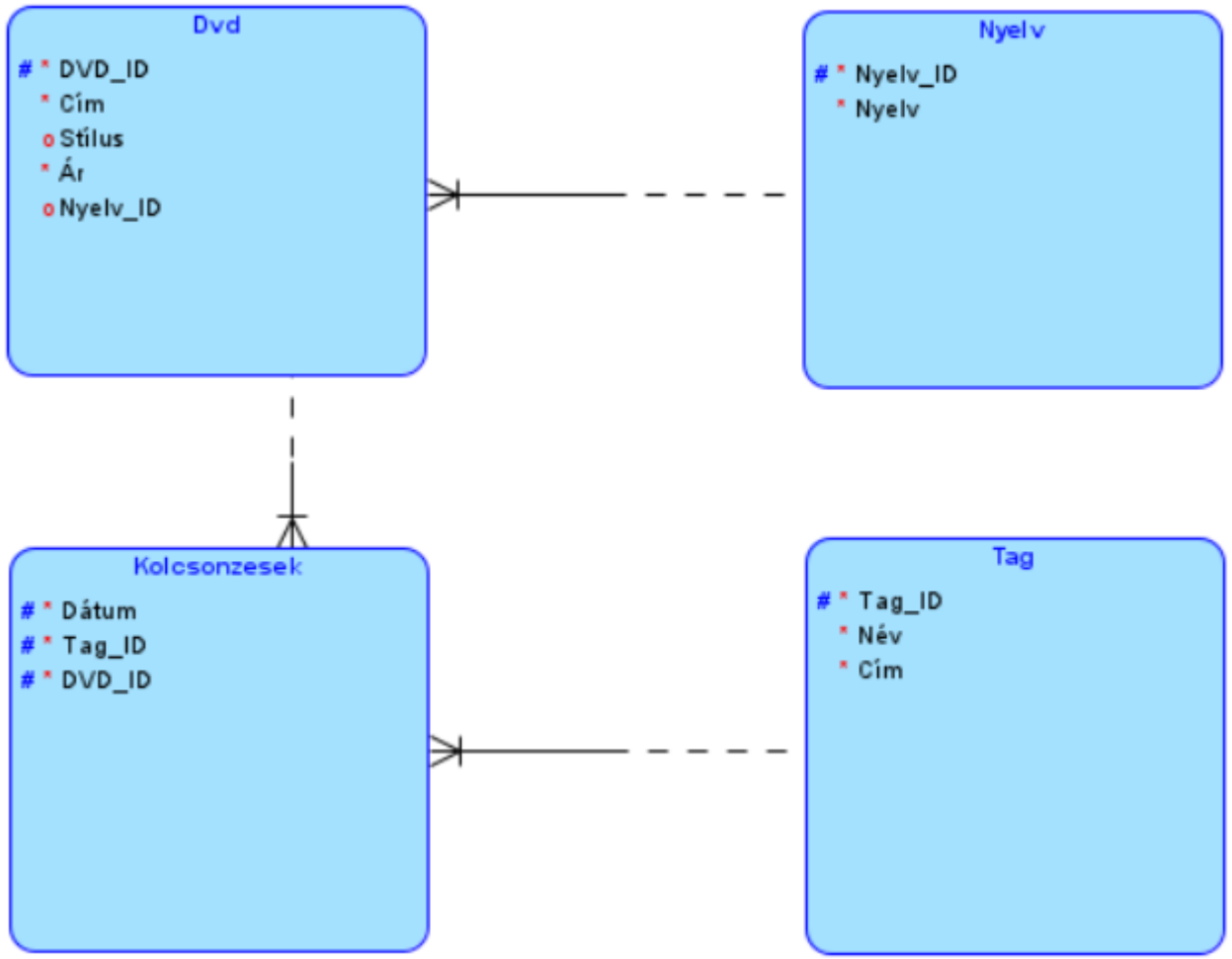
Adatbázisterv

Implementálás terve

BUDAPESTI CORVINUS EGYETEM

Logikai adatmodell

Logikai adatmodell* (Dvd adatbázis)



* Barker-fél jelölésrendszer

Fizikai adatmodell

Fizikai adatmodell

A fizikai adatmodell a konkrét hardver- és szoftverkörnyezetben történő implementálás tervét jelenti

A fizikai adatmodell kötelezően tartalmazza …

- ❑ A táblákat, kapcsolatokat, attribútumokat
- ❑ Az egyes attribútumok típusát, méretét, elsődleges és idegen kulcsokat
- ❑ A kényszereket

+ tartalmazhat minden olyan információt, amely az implementáshoz szükséges lehet (nézetek, indexek, partíciók, klaszterek, replikáció, tömörítés, titkosítás stb.)

BUDAPESTI CORVINUS EGYETEM

CASE-eszköz

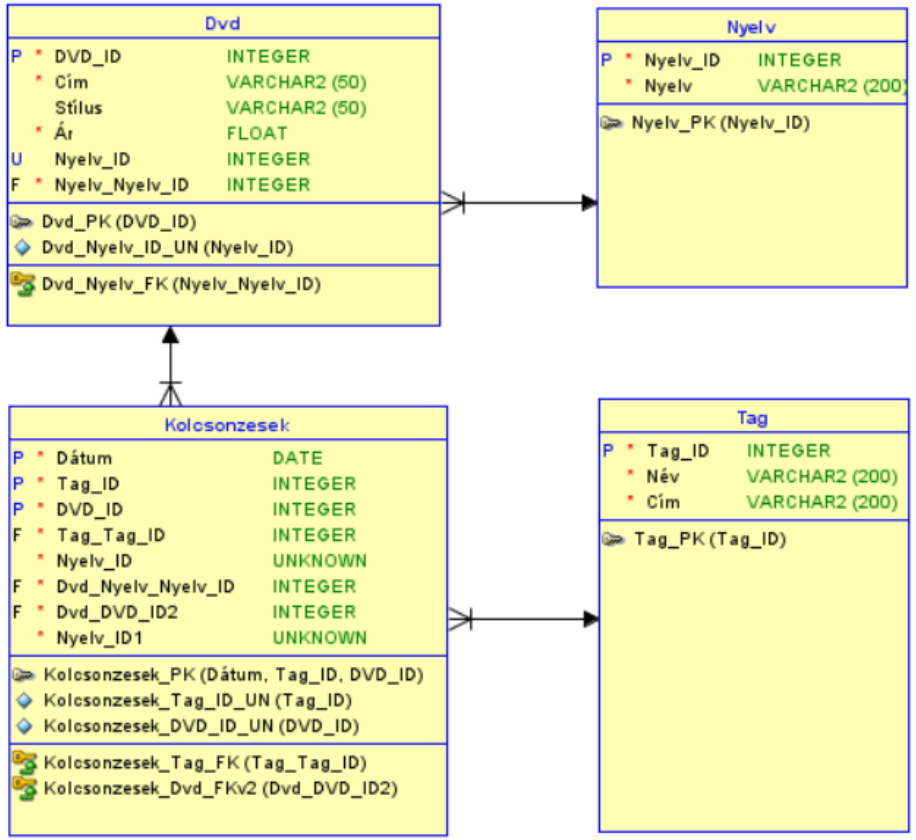
Adatmodellezés esetén olyan szoftver, amely segíti a modellezés teljes folyamatát

Tipikus funkciók

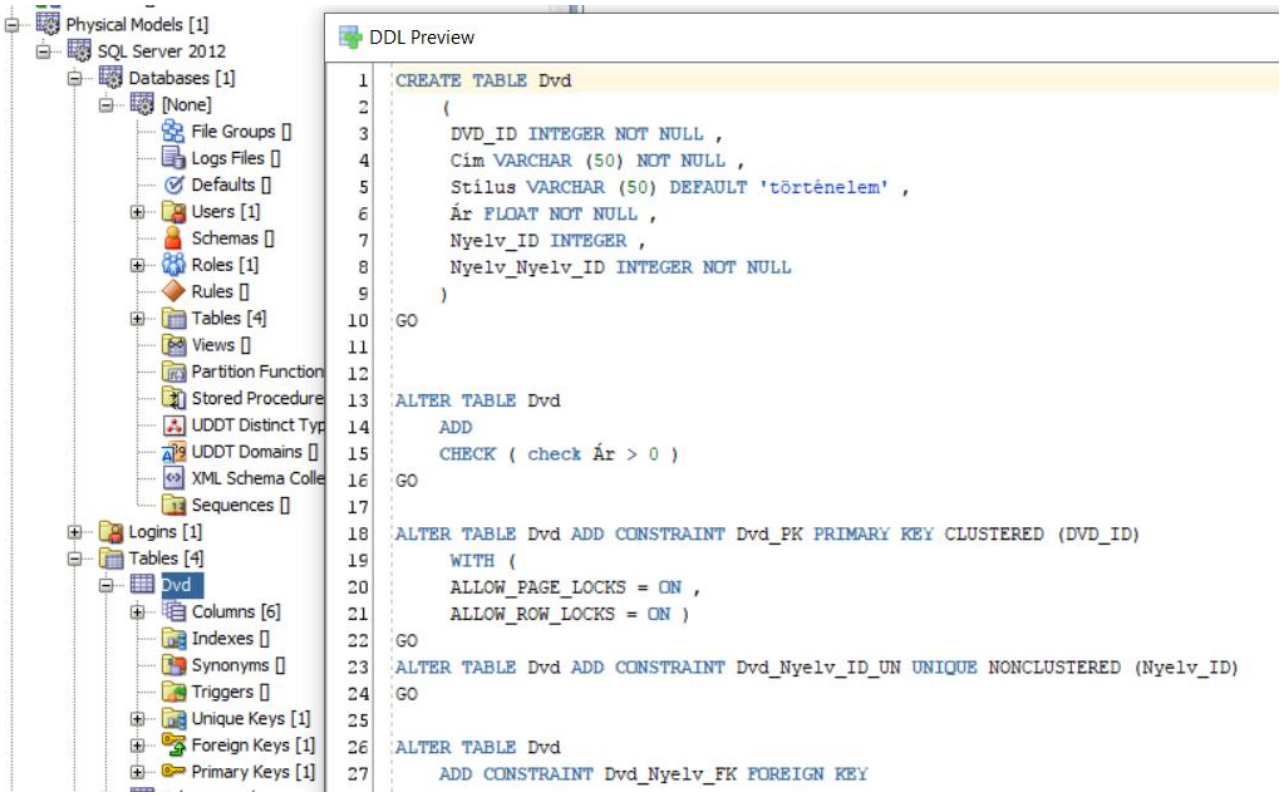
- ❑ Vizuális modellezés
- ❑ Adatstruktúrák tervezése
- ❑ Kódgenerálás
- ❑ Reverse engineering
- ❑ Dokumentáció
- ❑ Verziókezelés

Pl: Oracle SQL Data Modeler

A meglévő logikai modellből relációs modellt, abból pedig fizikait modellt generálni



Relációs modell (SQL Server 2012)

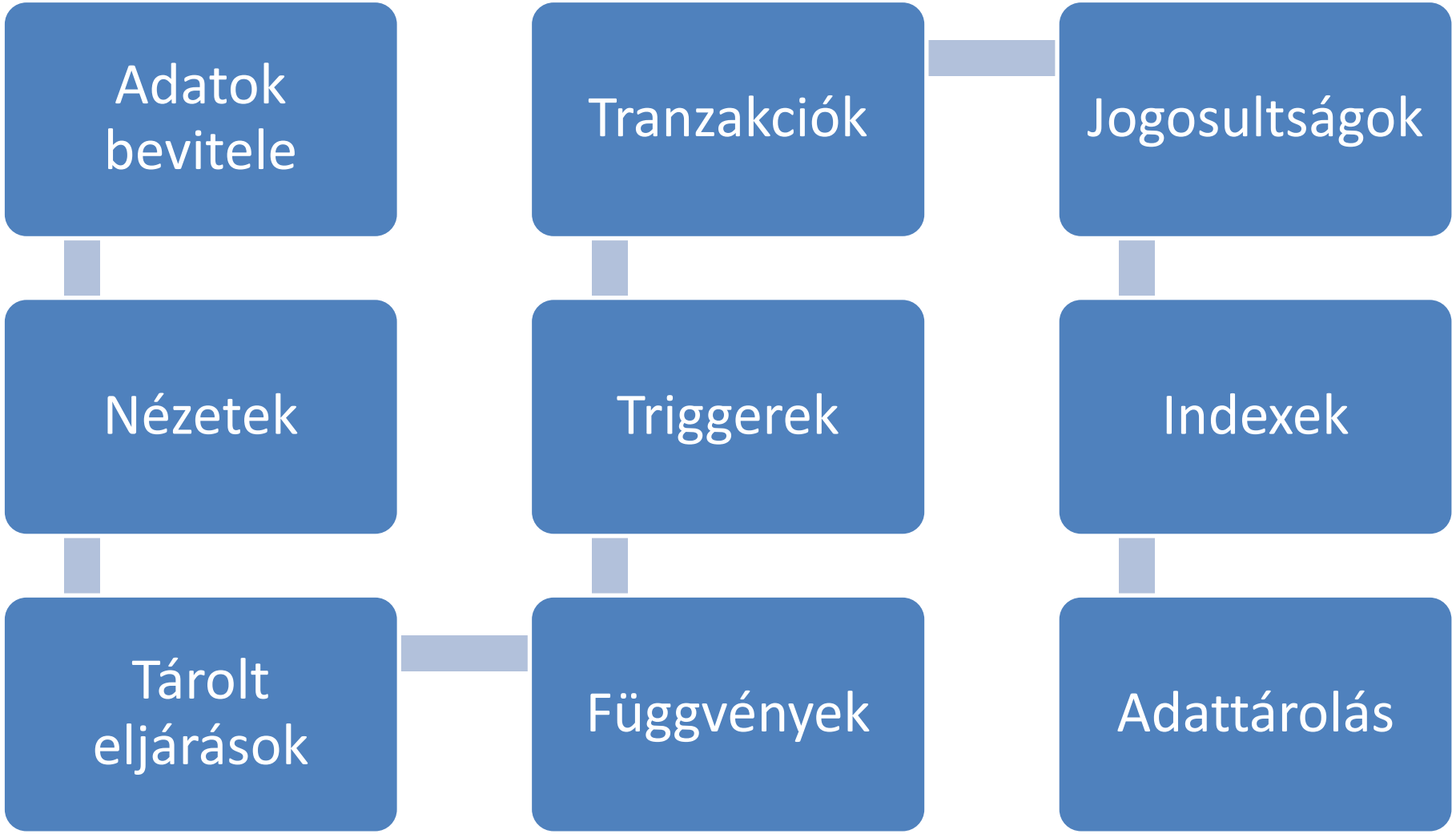


Fizikai modell + a Dvd tábla kódja

Adatbázisok fejlesztése

Készen van az adatbázis,
mit kell még csinálni?

Fejlesztési és konfigurálási feladatok



Adatbetöltés, adatbevitel

Adatbetöltés, adatbevitel

Hogyan kerülhetnek adatok az adatbázisba?

- ❑ Manuális adatbevitel
- ❑ Adatok importálása
- ❑ Adatok migrálása
- ❑ Adatok felvitele alkalmazásból
- ❑ ETL-folyamat
- ❑ Batch fájl futtatása

Nézetek

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

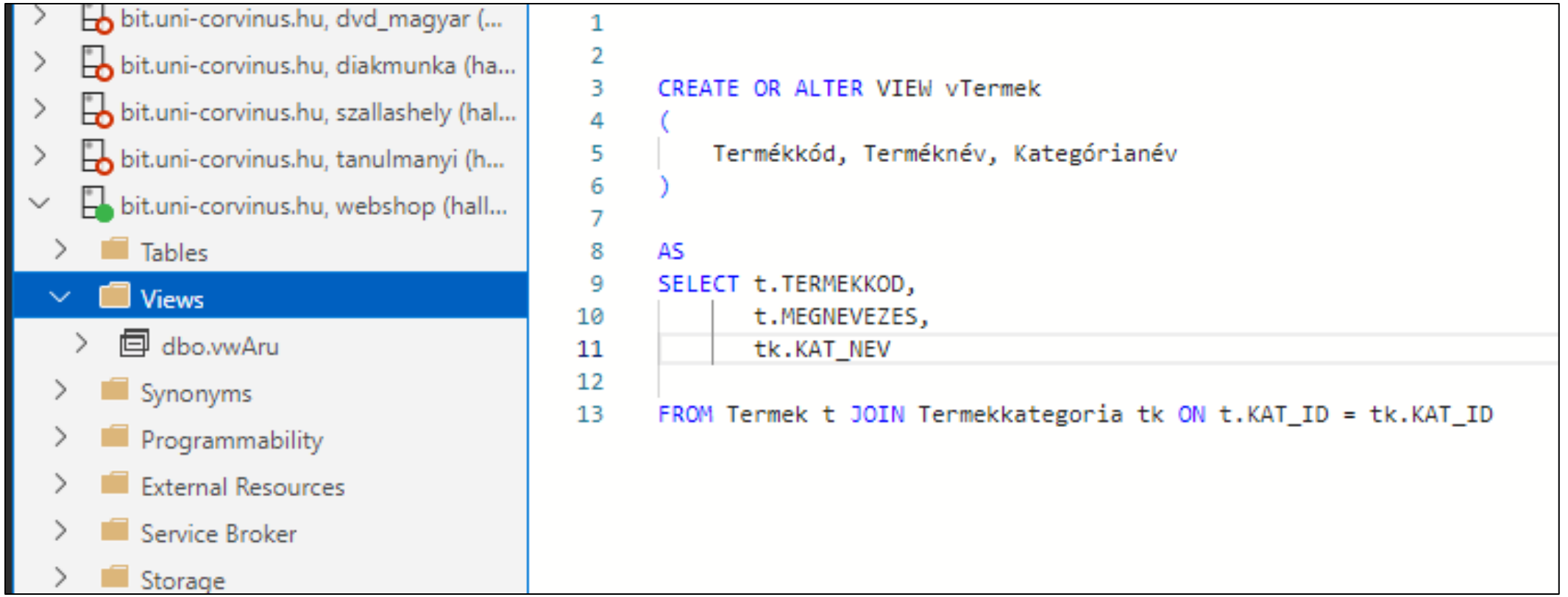
- Korlátozható az adatok elérése
- A bonyolultabb lekérdezések egyszerűbb formára hozhatók
- Az adatokat többféle nézőpontból szemlélhetjük
- Az adatfüggetlenség biztosítása

A Nézetek két fő típusa

Virtuális	Materializált
• Csak a lekérdezés tárolódik	• Az adatok is tárolásra kerülnek

Nézetek létrehozása: CREATE VIEW utasítás ➔ lásd gyakorlat

Nézetek – Azure Data Studio



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
```

BUDAPESTI CORVINUS EGYETEM

Tárolt eljárások

SQL kód vs. Alkalmazások

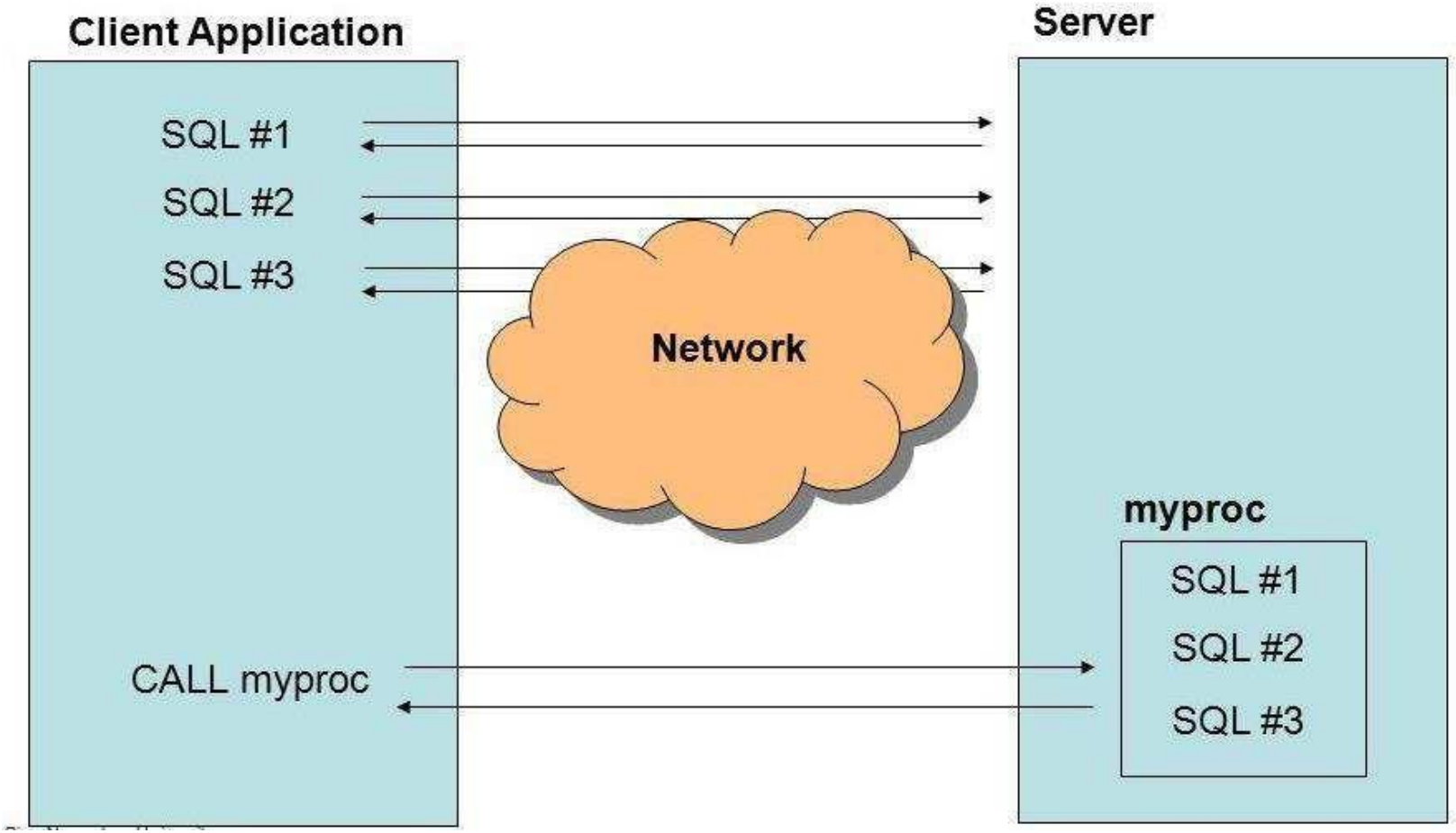
Mi a hátránya annak, ha az SQL kódot beépítjük a kliens alkalmazásba?

```
In [2]: import pymssql

In [3]: conn = pymssql.connect(server='sqlgyak.database.windows.net', user='hallgato', password='Password123', database='tanulmanyi')

In [5]: cursor = conn.cursor()
        cursor.execute('SELECT * FROM Termek')

In [9]: row = cursor.fetchone()
        while row:
                print (str(row[0]) + " " + str(row[1]))
                row = cursor.fetchone()
```

```
1 117
2 118
3 119
4 120
5 217
6 218
7 219
8 220
9 E.fsz.IV.
10 S.Asor.S3
11 E.fsz.I
12 116
13 VP 203.
14 E.2.238
15 E.3.332
16 116
```

Tesztelés?
Karbantartás?
Server-kliens kapcsolatok száma?
SQL kód újra felhasználása?
Jogosultságok?

A tárolt eljárások működése



A kép forrása: Bina Nusantara University

Tárolt eljárás (Stored procedure)

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

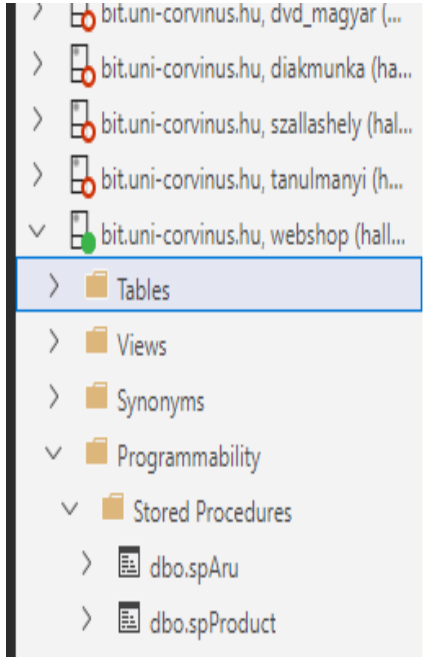
A tárolt eljárások főbb jellemzői

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

A tárolt eljárások előnyei

Hatékonyság	Fenntarthatóság	Biztonság	Üzleti logika elkülönítése
• Egyszerre több alkalmazás is használhatja őket • Csökken a szerver-kliens üzenetek száma	• A kódok egy központi helyen találhatók • A módosítás, tesztelés elkülönülhet a tárolt eljárást hívó alkalmazástól	• Használatukkal korlátozható a táblákhoz való hozzáférés • A hozzáférés biztosítása így nem a tárolt eljárást hívó alkalmazás feladata	• Az üzleti logika elkülönül a tárolt eljárást hívó alkalmazástól • Csökkenhet a kliens programok miatti adathibák száma

Tárolt eljárások az MS SQL-ben




```
1
2
3   CREATE OR ALTER PROC spTermekRendelesek
4   @termekkod NVARCHAR(255)
5
6   AS
7   BEGIN
8       SELECT *
9       FROM Rendeles_tetel
10      WHERE TERMEKKOD = @termekkod
11  END
```

```
CREATE PROCEDURE procedure_name

-- paraméterek megadása

AS

BEGIN

-- SQL utasítások

END
```

BUDAPESTI CORVINUS EGYETEM

Függvények

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

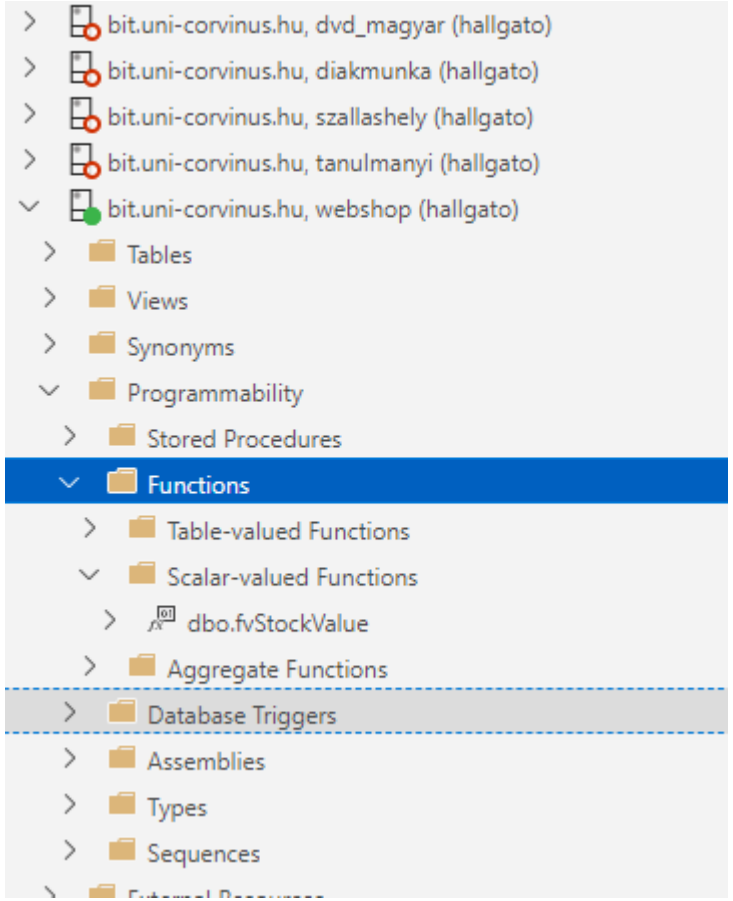
- Input paramétereket, SQL-utasításokat, valamint különböző algoritmikus szerkezeteket is tartalmazhatnak (elágazás, ciklus)
- Az adatbázis szerveren tárolódnak
- Futtatásuk jogosultságokhoz köthető
- Felhasználhatók SQL-utasításokban, pl: SELECT utasításban

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




```
CREATE FUNCTION function_name
(
-- paraméterek megadása
)
RETURNS adattípus

AS

BEGIN

-- SQL utasítások

RETURN érték

END
```

BUDAPESTI CORVINUS EGYETEM

Triggerek

Triggerek

Olyan speciális eljárások, amelyek bizonyos események bekövetkezéséhez köthetően automatikusan végrehajtódnak.

- Típusai:
  - DML triggerek* – adatmanipuláció esetén futnak le (pl. INSERT, DELETE)
  - DDL triggerek – adatdefiníció esetén futnak le (pl. CREATE, DROP)
  - Logon triggerek – bejelentkezéskor futnak le
- Alkalmazásuk:
  - Kényszerek, üzleti szabályok definiálása
  - Hivatkozási integritás biztosítása
  - Logolás, nyomkövetés

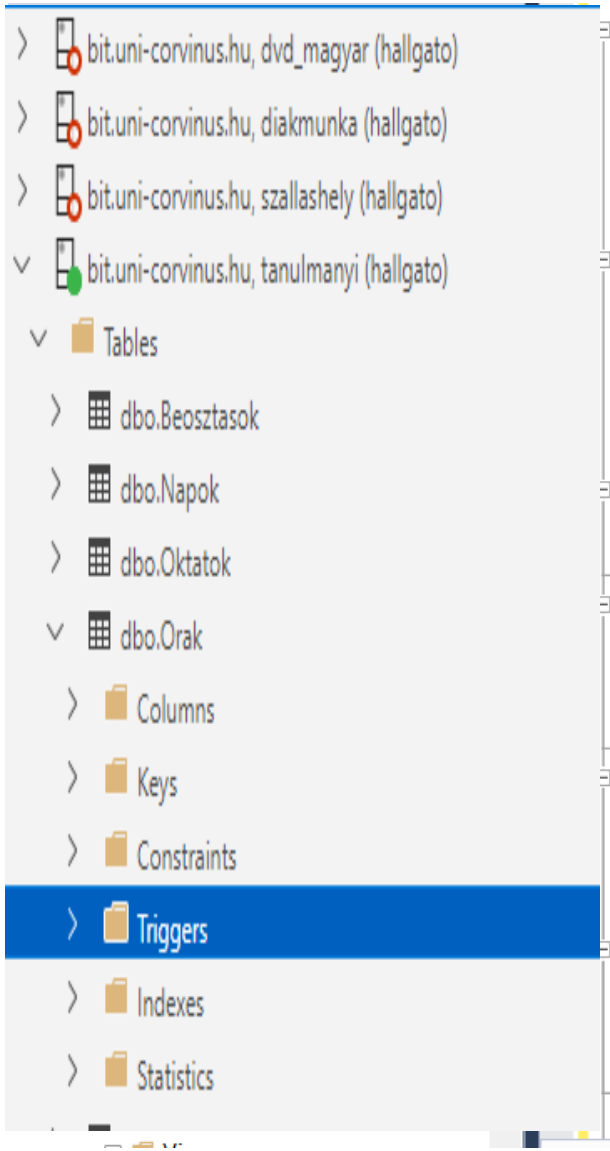
*Csak a DML triggerekkel foglalkozunk

DML Triggerek létrehozása MS SQL-ben

```
CREATE TRIGGER triggernév
ON táblanév
FOR | AFTER | INSTEAD OF
INSERT | UPDATE | DELETE
AS
BEGIN
   -- SQL utasítások
END
```

Pl: egy oktatónak maximum 10 órája lehet

Triggerek - példa




```
CREATE TRIGGER tg_max_ora
ON ORAK
INSTEAD OF INSERT
AS
BEGIN
   DECLARE @tanar int
   DECLARE @oraszam int
   DECLARE @maxoraszam int = 10
     SELECT @tanar = tanar
   FROM inserted
   SELECT @oraszam = COUNT(*)
   FROM Orak
   WHERE tanar = @tanar
   IF @oraszam >= @maxoraszam
       PRINT 'Nem lehet több órája'
   ELSE
     INSERT INTO Orak
     SELECT i.*
     FROM inserted i
END
```

Az inserted tábla tartalmazza az új vagy módosult sorok másolatát

Triggerek – előnyök és hátrányok

- ☺ Viszonylag egyszerű kód
- ☺ Sokoldalú felhasználás
- ☺ Meghívhatnak tárolt eljárásokat, függvényeket
- ☺ Meghívhatnak külső kódot
- ☺ Támogatják a rekurziót
- ☺ Egymásba ágyazhatók

- ☹ Performancia
- ☹ Nehézkes tesztelés és hibakeresés
- ☹ Biztonsági problémák
- ☹ A kliens alkalmazások számára nem láthatók

BUDAPESTI CORVINUS EGYETEM

Tranzakciók

Probléma - átutalás

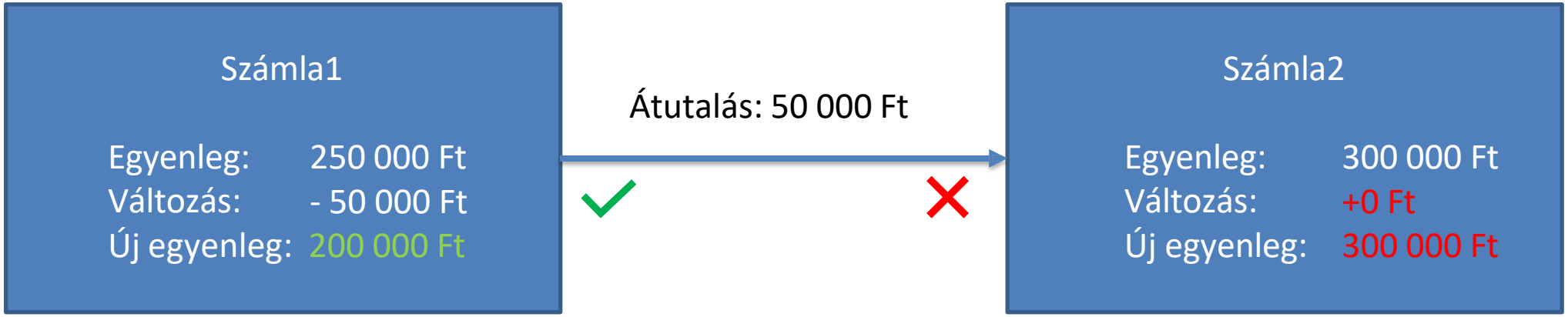
Az átutalás két fő lépése:

- Levonni az összeget az 1. számláról
- Jóváírni az összeget a 2. számlán

Mi történhet, ha ezt a két lépést egyesével (egymás után és egymástól függetlenül) hajtjuk végre?

Probléma – átutalás (folytatás)

Az átutalás folyamata megszakadhat pl. az első lépés után



A folyamat megszakadása esetén inkonzisztens adatok lehetnek az adatbázisban

Probléma – átutalás (megoldás)

Kezeljük egyetlen logikai egységként az átutalás 2 fő lépését!

Számla1

Egyenleg: 250 000 Ft
Változás: - 50 000 Ft
Új egyenleg: 200 000 Ft

Átutalás: 50 000 Ft

Számla2

Egyenleg: 300 000 Ft
Változás: +50 000 Ft
Új egyenleg: 350 000 Ft

Ha nem lép fel hiba, akkor az összetett lépéssorozat rendben végrehajtódik

Számla1

Egyenleg: 250 000 Ft
Változás: - 50 000 Ft
Új egyenleg: 200 000 Ft

Átutalás: 50 000 Ft

Számla2

Egyenleg: 300 000 Ft
Változás: +0 Ft
Új egyenleg: 300 000 Ft

Számla1

Egyenleg: 250 000 Ft
Változás: -0 Ft
Új egyenleg: 250 000 Ft

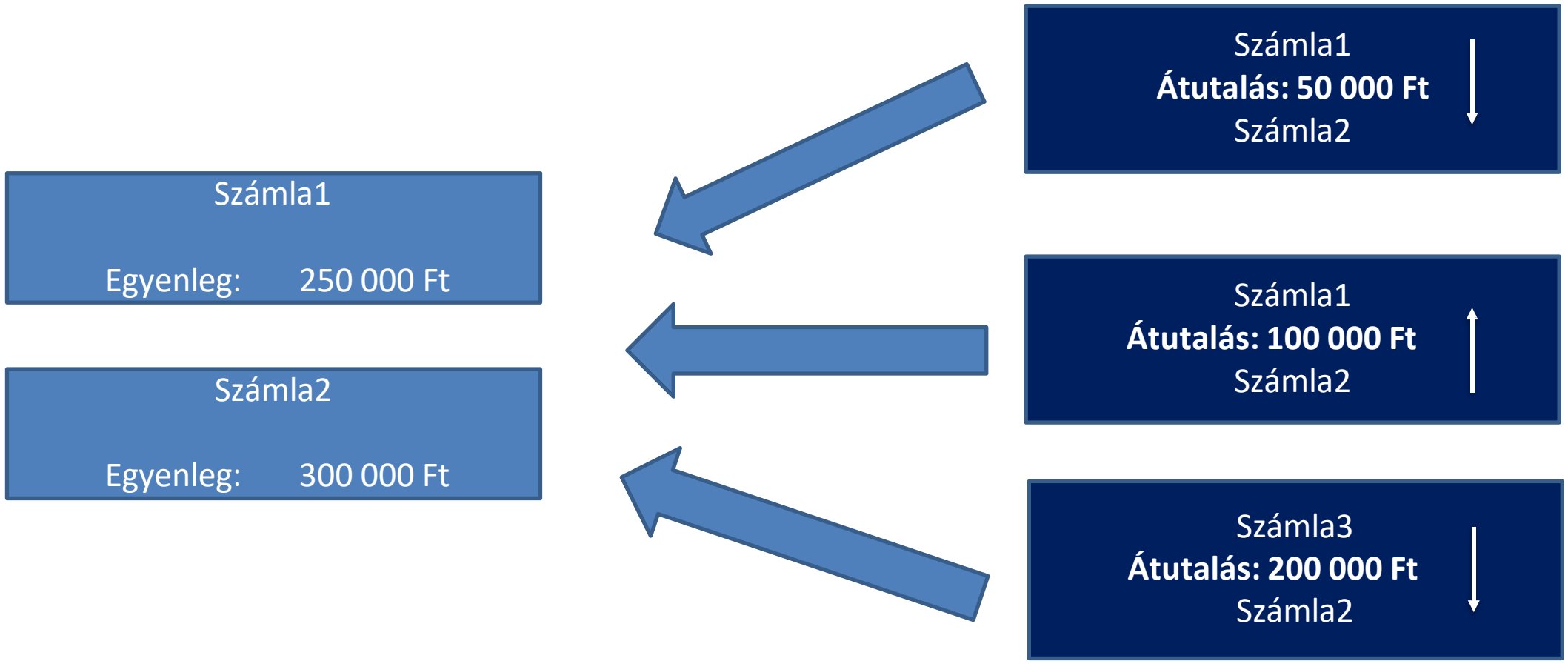
Számla2

Egyenleg: 300 000 Ft
Változás: +0 Ft
Új egyenleg: 300 000 Ft

Ha valahol megszakad a folyamat, akkor visszaáll az eredeti állapot

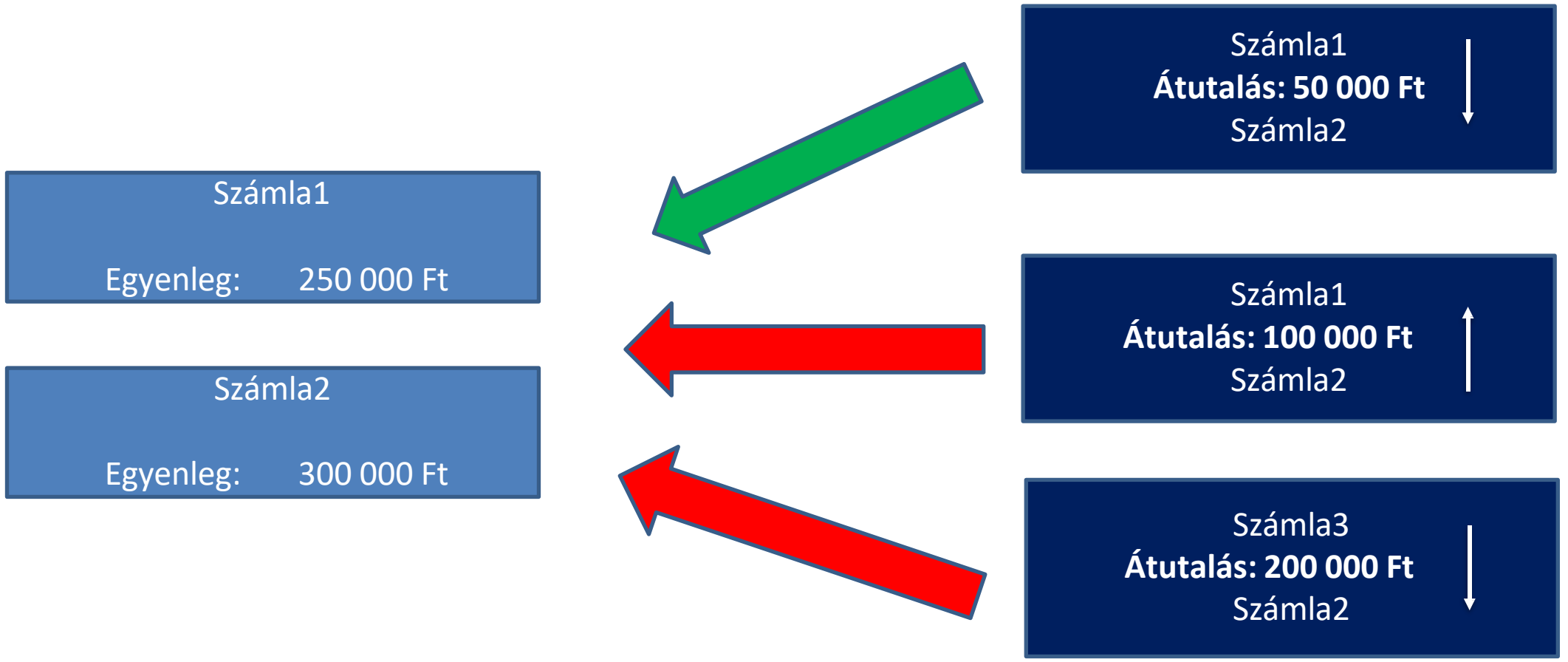
Probléma2 – átutalás

Mi történik, ha egyszerre több átutalás indul, amely érinti valamelyik számlát?



Probléma2 – átutalás (megoldás)

Korlátozni (szabályozni) kell az egyidejű hozzáférést!



Tranzakció

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)

Kép forrása: https://ask.careers/news/pnb-customers-to-be-charged-after-5-atm-transactions-starting-october/

Tranzakció tulajdonságok - ACID

A kép forrása: https://dev.to/princessanjana1996/acid-properties-in-databases-43aa

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 korlátozza az adatok egyidejű elérését a tranzakciók számára.

- 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 (pl: sor, tábla) és többféle módon (pl: kizárólagos, megosztott)

Zárolás az MS SQL-ben

Fontosabb zárolható erőforrások

Erőforrás	Leírás
RID	Egy sor
Key	Az indextábla egy sora
Page	Egy oldal (fizikai tárolási egység)
Extent	Több (8 db) oldal
Table	Egy tábla
DB	Az egész adatbázis
Application	Alkalmazás-specifikus erőforrások
File	Adatbázis fájl
Metadata	Katalógus információk
Object	Adatbázis objektumok
Xact	Tranzakció erőforrásai

Zárolási módok

Mód	Betűjel
Shared	S
Update	U
Exclusive	X
Intent	I
Schema	Sch
BU	Bulk Update

BUDAPESTI CORVINUS EGYETEM

Zárolási kompatibilitás az MS SQL-ben

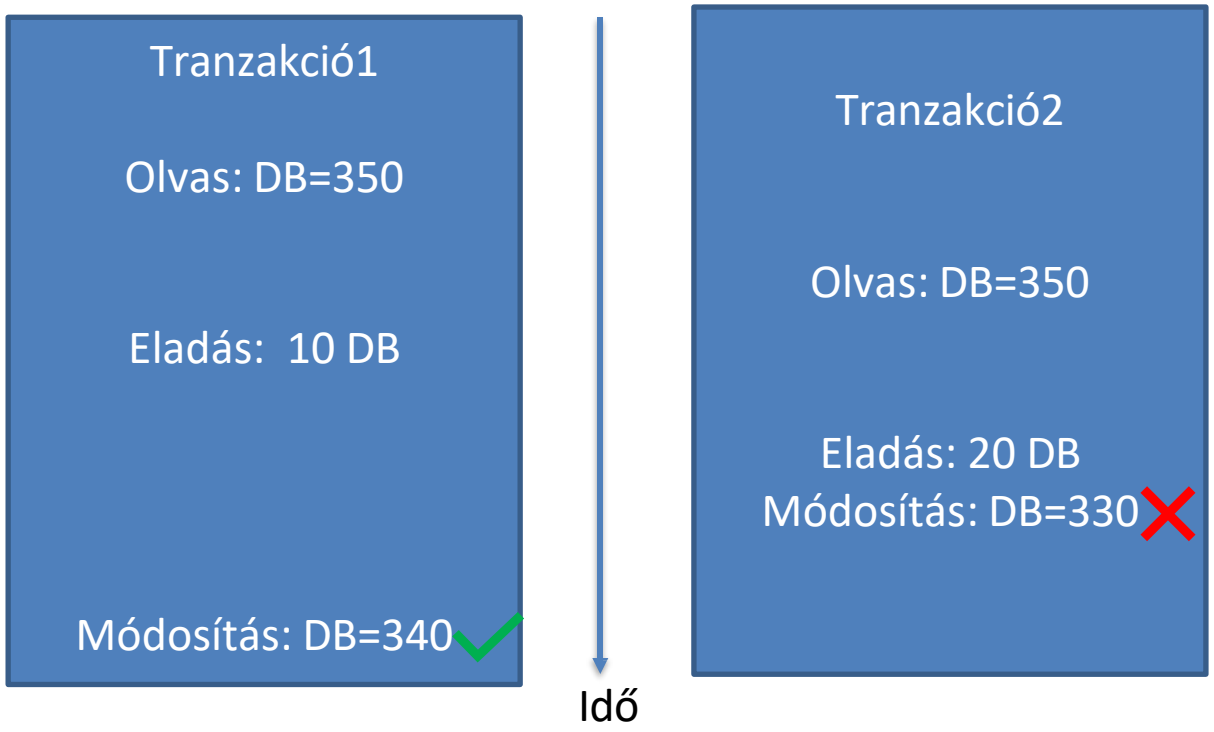
Existing granted mode	IS	S	U	IX	SIX	X
Requested mode						
Intent shared (IS)	Yes	Yes	Yes	Yes	Yes	No
Shared (S)	Yes	Yes	Yes	No	No	No
Update (U)	Yes	Yes	No	No	No	No
Intent exclusive (IX)	Yes	No	No	Yes	No	No
Shared with intent exclusive (SIX)	Yes	No	No	No	No	No
Exclusive (X)	No	No	No	No	No	No

Meglévő zárolás esetén egy új tranzakció zárolási igénye csak akkor teljesülhet, ha az kompatibilis a meglévő zárolással. Ellenkező esetben az új tranzakciónak várakoznia kell.

Egyidejű (konkurens) tranzakciók kezelése

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.

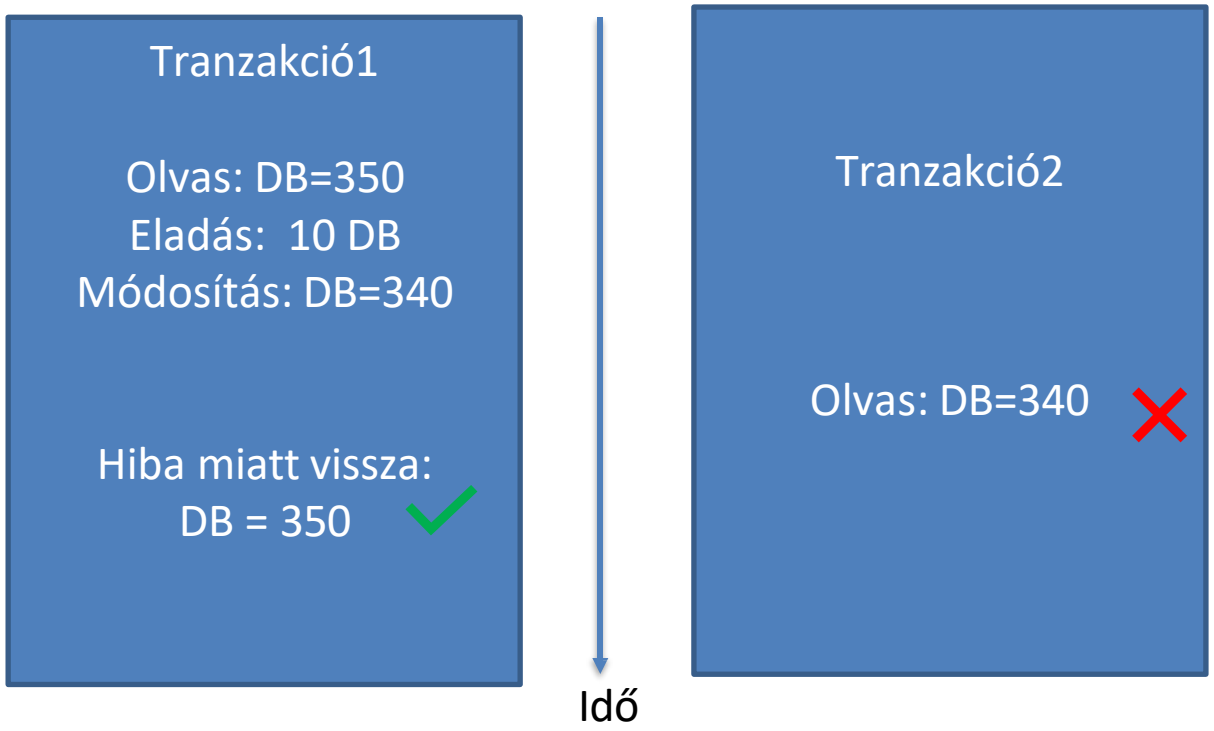
KÉSZLET		
ID	Termék	DB
1	Tégla	350
2	Cement 50 kg	50



Egyidejű tranzakciók kezelése (folyt.)

„Piszkos” adatok olvasása (dirty reads)
Egy nem véglegesített tranzakció adatait olvassuk. Az adat azonban még változhat a tranzakció végrehajtása során.

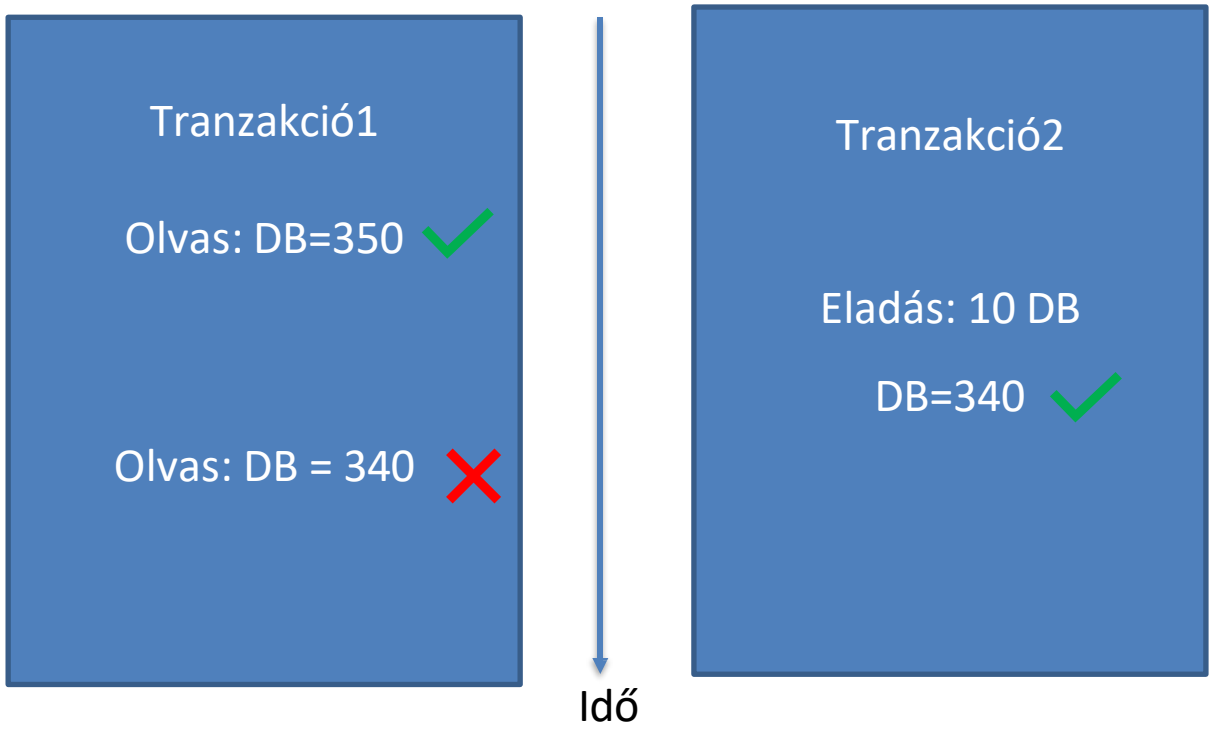
KÉSZLET		
ID	Termék	DB
1	Tégla	350
2	Cement 50 kg	50



Egyidejű 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.

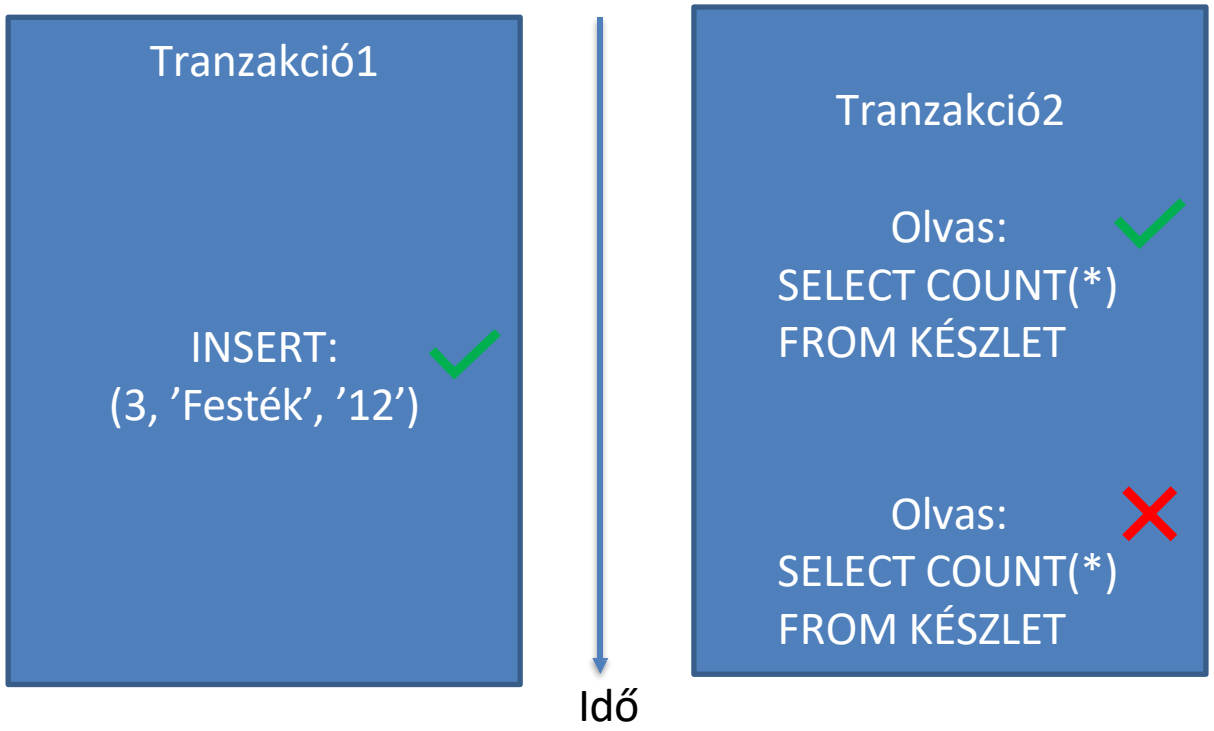
KÉSZLET		
ID	Termék	DB
1	Tégla	350
2	Cement 50 kg	50



Egyidejű tranzakciók kezelése (folyt.)

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

KÉSZLET		
ID	Termék	DB
1	Tégla	350
2	Cement 50 kg	50



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)
- ❑ Repetable read: az olvasott adatot nem módosíthatja más tranzakció
- ❑ Seriazable: 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 lévők (kevésbé szigorú szintek) korlátozásait is.

BUDAPESTI CORVINUS EGYETEM

Konkurencia problémák és izolációs szintek

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

SQL SERVER tranzakciós 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

SQL Server tranzakciós módok (folyt)

- **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

SELECT (ha táblát is érint)	CREATE	INSERT	UPDATE
DROP	ALTER TABLE	TRUNCATE TABLE	DELETE
MERGE	GRANT	REVOKE	FETCH

Explicit tranzakciók megvalósítása SQL-ben

```
SELECT COUNT(*)    --16
FROM Termek

BEGIN TRANSACTION t1

INSERT INTO Termek VALUES(20, 'Húszas terem')
SAVE TRANSACTION s1

INSERT INTO Termek VALUES(30, 'Harmincas terem')

ROLLBACK TRANSACTION s1
SELECT COUNT(*)    --17
FROM Termek

INSERT INTO Termek VALUES(30, 'Harmincas terem')
COMMIT

SELECT COUNT(*)    --18
FROM Termek
```

Implicit tranzakciók megvalósítása SQL-ben

```
SET IMPLICIT_TRANSACTIONS ON

SELECT COUNT(*)   --16
FROM Termek

INSERT INTO Termek VALUES(20, 'Húszas terem')
ROLLBACK

SELECT COUNT(*)   --16
FROM Termek

INSERT INTO Termek VALUES(20, 'Húszas terem')
COMMIT

SELECT COUNT(*)   --17
FROM Termek
```

Jogosultságok

Jogosultságokkal kapcsolatos fogalmak

- Azokat az felhasználói fiókokat, amelyekkel a felhasználók hozzáférhetnek az SQL-szerverhez, **LOGIN**-oknak nevezzük
- Azokat az identitásokat, akik számára jogosultságok megadhatók, SECURITY **PRINCIPAL**-oknak („biztonsági résztvevő”) nevezzük, pl: felhasználó, szerepkör – akik kapják a jogosultságokat
- Azokat az objektumokat, amelyekhez a jogosultságok rendelhetők, **SECURABLE**-knek („biztosítandó”) nevezzük, pl: szerver, adatbázis – amihez jogok rendelhetők
- Azokat a rekordokat, amelyek az SQL-szerveren kívüli erőforrásokhoz való csatlakozáshoz szükséges hitelesítési információkat tartalmazzák, **CREDENTIAL**-oknak („meghatalmazás”) nevezzük. Egy ilyen rekord általában nevet és jelszót tartalmaz.

Jogosultságok adása, visszavonása és megtagadás (SQL-szerver)

AUTHORIZATION PERMISSION ON SECURABLE TO PRINCIPAL;

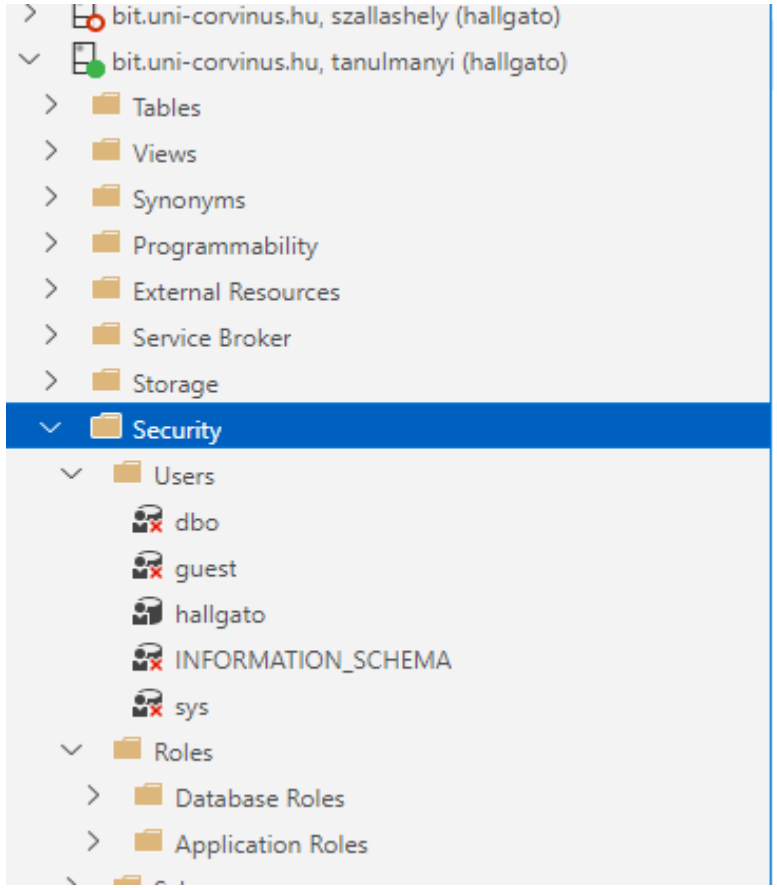
- AUTHORIZATION (engedély) lehet GRANT, REVOKE és DENY (jog adása, visszavonása és megtagadása)
- PERMISSION (a konkrét jogosultság), több, mint 200 féle, pl: SELECT, EXECUTE, UPDATE
- SECURABLE lehet szerver, szerver objektum, adatbázis, adatbázis objektum
- PRINCIPAL lehet LOGIN, felhasználó vagy szerepkör

Pl: GRANT UPDATE ON OBJECT::Product TO Ted;
(UPDATE jog adása Ted felhasználó számára a Product táblához)

Jogosultságok (SQL-szerver)

Az SQL-szerver jogosultságok megadhatók

- Szerver szinten –
  Login-ok és szerver szerepkörök által (logins, server roles)
- Adatbázis szinten –
  Adatbázis felhasználók és adatbázis szerepkörök által
  (database users, database roles)



Szerver-szintű szerepkörök

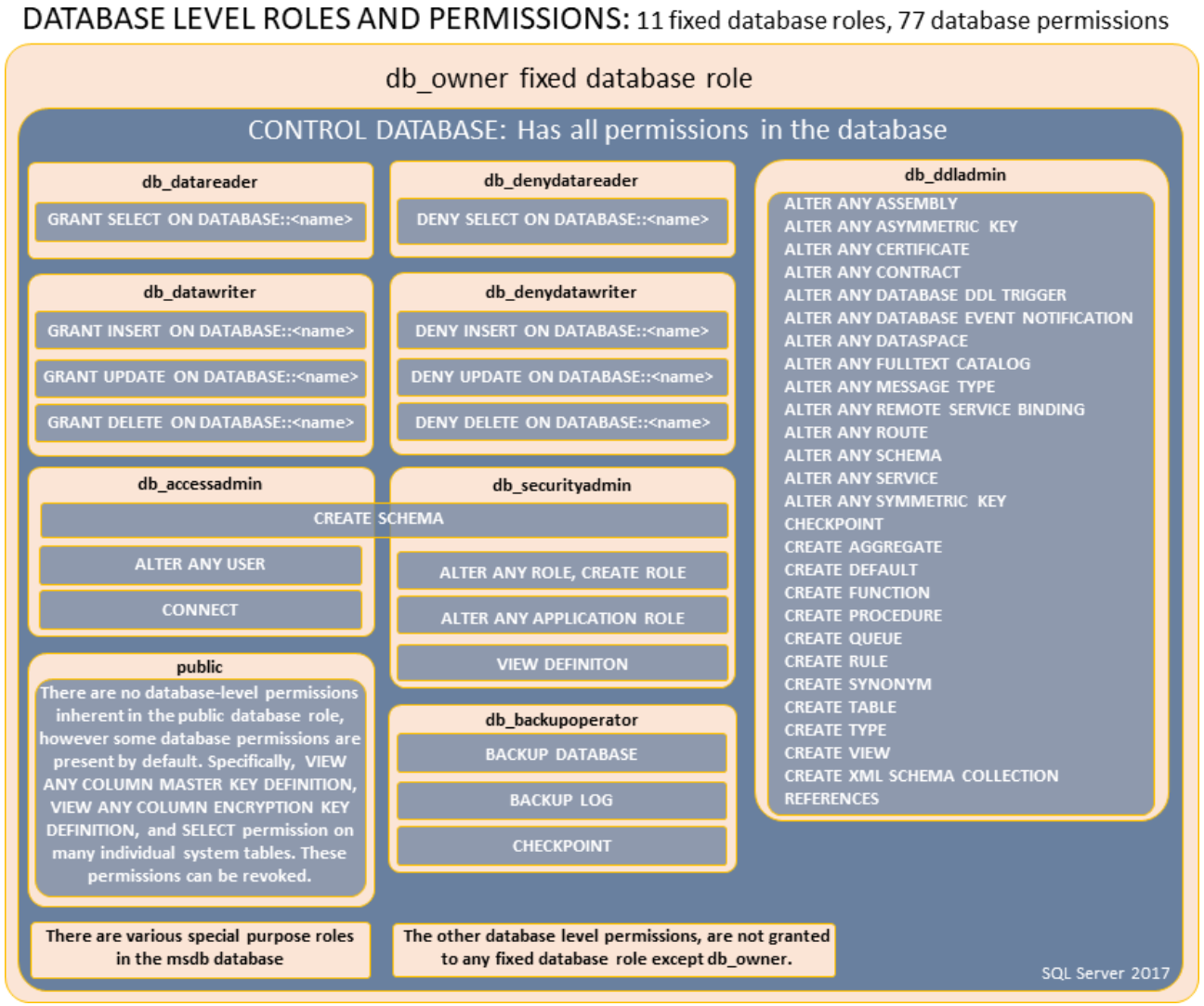
Jogosultság	Rövid leírás
Sysadmin	Teljes joggal rendelkezik a szerveren
Serveradmin	Módosíthatja a szerver konfigurációt
Securityadmin	Szerver-szintű jogosultságokat kezelhet. Ha van hozzáférése adatbázisokhoz, akkor ott adatbázis-szintű jogokat is adhat vagy megtagadhat, elvehet.
Processadmin	Leállíthatja a futó folyamatokat (processzeket)
Setupadmin	Linked szervereket adhat hozzá, vagy törölhet
Diskadmin	A lemezen lévő adatbázis-fájlokat menedzselheti
Dbcreator	Adatbázisokat hozhat létre, módosíthat, törölhet
Public	Alapértelmezett jog

Adatbázis-szintű szerepkörök

Jogosultság	Rövid leírás
Db_owner	(Majdnem) teljes joggal rendelkezik az adatbázison
Db_securityadmin	Módosíthatja az egyedi szerepkörök (custom role) tagságát és jogosultságait
Db_accessadmin	Az adatbázis elérését engedélyezheti vagy visszavonhatja a LOGIN-ok számára
Db_backupoperator	Biztonsági mentést készíthet az adatbázisról
Db_ddladmin	Tetszőleges DDL parancsot kiadhat
Db_datawriter	Módosíthatja a felhasználói táblákat
Db_datareader	Olvashatja a felhasználói táblákat
Db_denydatawriter	Nem módosíthatja a felhasználói tábákat
Db_denydatareader	Nem olvashatja a felhasználói táblákat

BUDAPESTI CORVINUS EGYETEM

Adatbázis-szintű szerepkörök

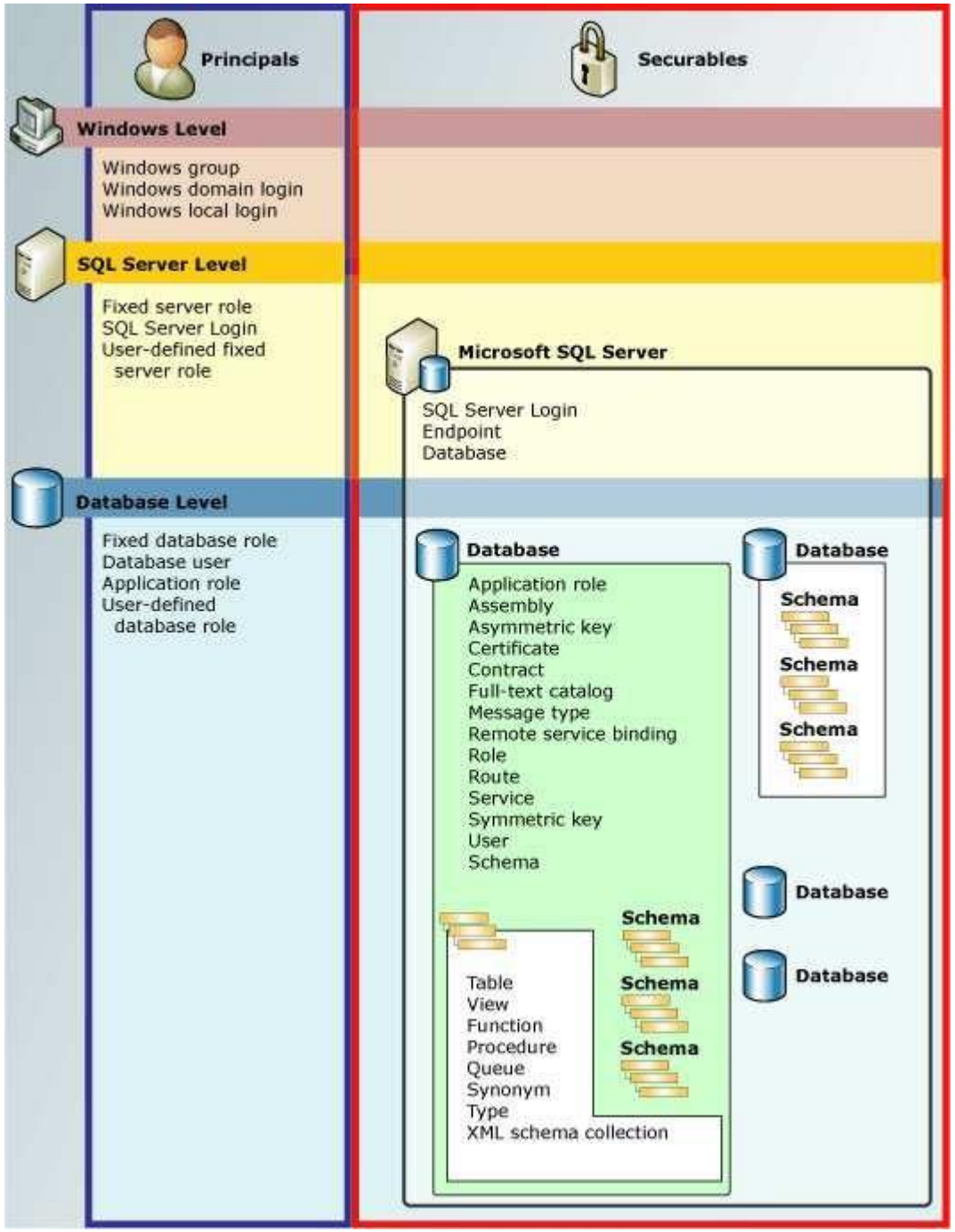


Tábla jellegű objektumok jellemző jogosultságai (táblák, nézetek, tábla-értékű függvények)

Jogosultság	Elvégezhető művelet (korlátozható oszlopokra is)
SELECT	olvasás táblázatból, nézetből
INSERT	adatbevitel táblázatba, nézetbe
DELETE	Sor(ok) törlése táblázatból, nézetből
UPDATE	adatok módosítása táblázatban, nézetben
REFERENCES	idegen kulccsal való hivatkozás táblázatra
ALL	minden művelet

BUDAPESTI CORVINUS EGYETEM

Jogosultság-hierarchia



Jogosultságok lekérdezése – szerver-szinten

```
USE master
-- szerver-szinten
SELECT pr.principal_id,
    pr.name, pr.type_desc,
        pe.state_desc,
    pe.permission_name
FROM sys.server_principals AS pr
JOIN sys.server_permissions AS pe
ON pe.grantee_principal_id = pr.principal_id
```

Results | Messages

	principal_id	name	type_desc	state_desc	permission_name
1	1	sa	SQL_LOGIN	GRANT	CONNECT SQL
2	2	public	SERVER_ROLE	GRANT	VIEW ANY DATABASE
3	101	##MS_SQLResourceSigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	VIEW ANY DEFINITION
4	102	##MS_SQLReplicationSigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	AUTHENTICATE SERVER
5	102	##MS_SQLReplicationSigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	VIEW ANY DEFINITION
6	102	##MS_SQLReplicationSigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	VIEW SERVER STATE
7	103	##MS_SQLAuthenticatorCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	AUTHENTICATE SERVER
8	105	##MS_PolicySigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	CONTROL SERVER
9	105	##MS_PolicySigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	VIEW ANY DEFINITION
10	106	##MS_SmoExtendedSigningCertificate##	CERTIFICATE_MAPPED_LOGIN	GRANT	VIEW ANY DEFINITION
11	257	##MS_PolicyTsqlExecutionLogin##	SQL_LOGIN	GRANT	CONNECT SQL
12	257	##MS_PolicyTsqlExecutionLogin##	SQL_LOGIN	GRANT	VIEW ANY DEFINITION

Query executed successfully. (local)\sqlexpress (13.0 SP2) GROUP\T\51282 (53) master 00:00:00 34 rows

Jogosultságok lekérdezése – adatbázis-szinten

```
 USE master
--adatbázis-szinten

SELECT DISTINCT pr.principal_id, pr.name, pr.type_desc,
    pr.authentication_type_desc, pe.state_desc,
pe.permission_name
FROM sys.database_principals AS pr
JOIN sys.database_permissions AS pe
    ON pe.grantee_principal_id = pr.principal_id;
```

Results | Messages

	principal_id	name	type_desc	authentication_type_desc	state_desc	permission_name
1	0	public	DATABASE_ROLE	NONE	GRANT	EXECUTE
2	0	public	DATABASE_ROLE	NONE	GRANT	SELECT
3	0	public	DATABASE_ROLE	NONE	GRANT	VIEW ANY COLUMN ENCRYPTION
4	0	public	DATABASE_ROLE	NONE	GRANT	VIEW ANY COLUMN MASTER KEY
5	1	dbo	SQL_USER	INSTANCE	GRANT	CONNECT
6	2	guest	SQL_USER	NONE	GRANT	CONNECT
7	5	##MS_PolicyEventProcessingLogin##	SQL_USER	INSTANCE	GRANT	CONNECT
8	5	##MS_PolicyEventProcessingLogin##	SQL_USER	INSTANCE	GRANT	EXECUTE
9	6	##MS_AgentSigningCertificate##	CERTIFICATE_MAPPED_USER	NONE	GRANT	CONNECT
10	6	##MS_AgentSigningCertificate##	CERTIFICATE_MAPPED_USER	NONE	GRANT	EXECUTE

Köszönöm a figyelmet!