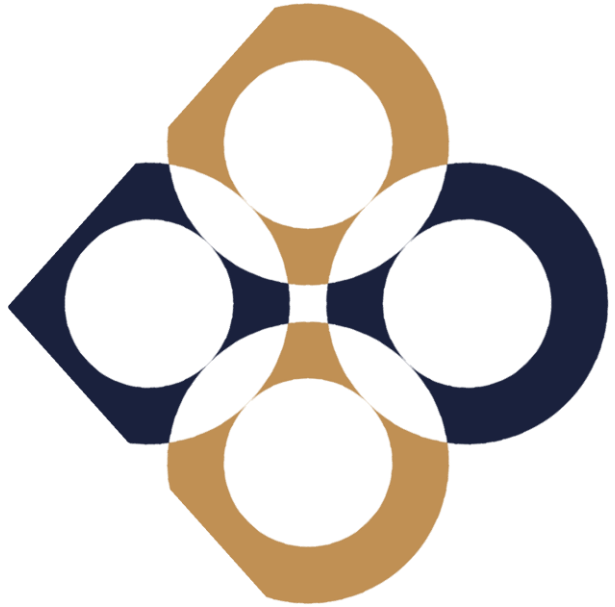
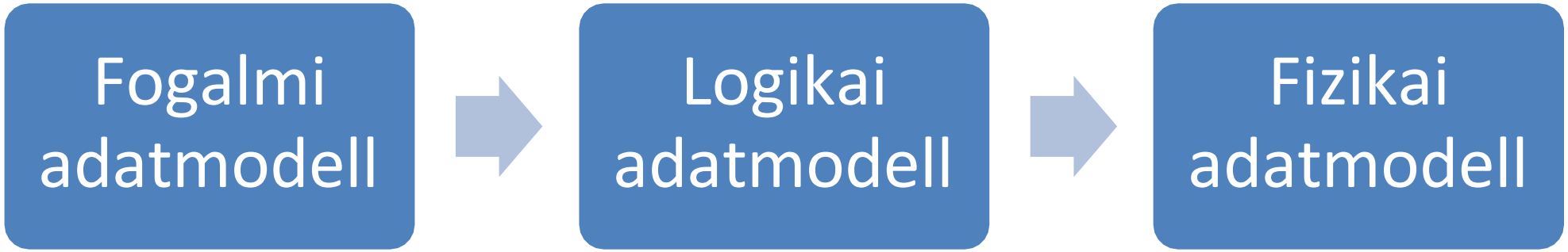
BUDAPESTI CORVINUS EGYETEM

# Adatbázisok előadás 04

Fizikai adatbázisterv
Adatbázisok fejlesztése



# A tervezés lépései




Egyed – kapcsolat modell (ER-modell)

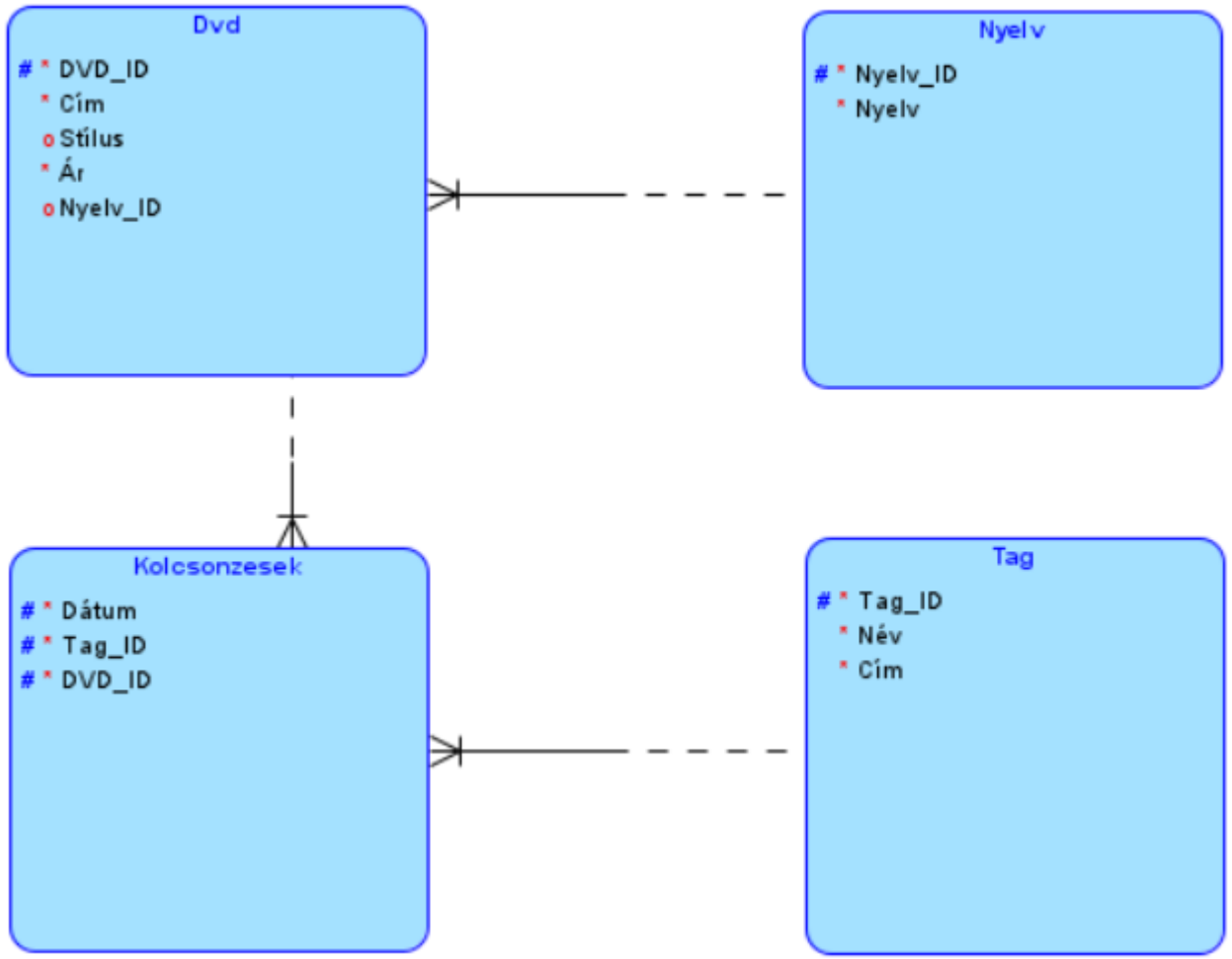
Adatbázisterv

**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

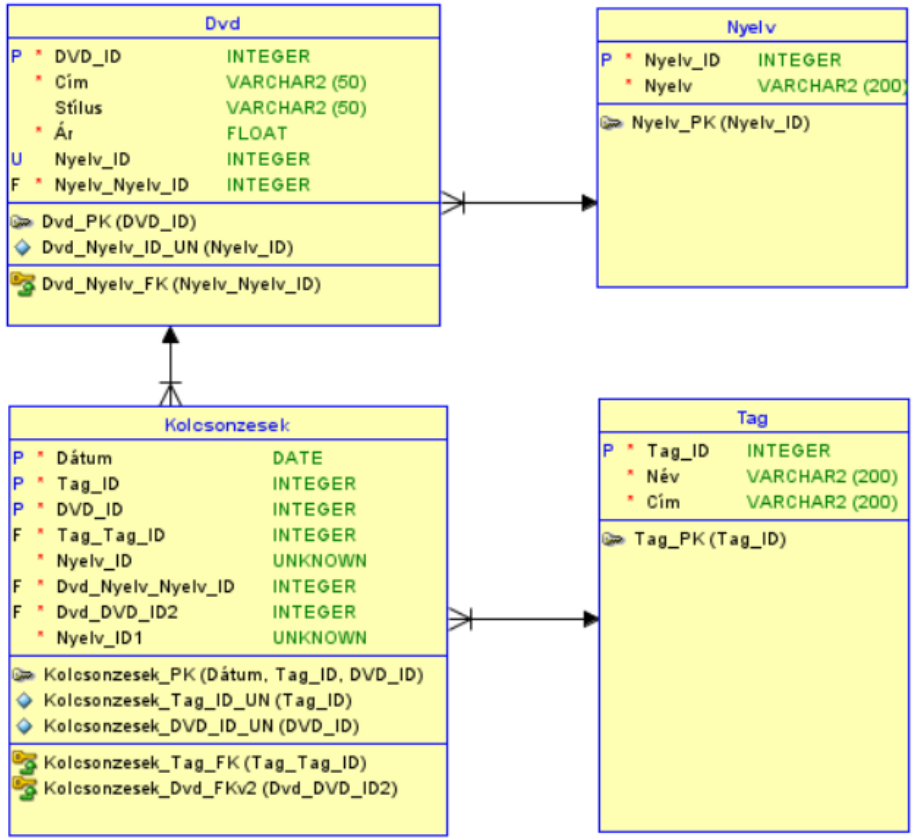
Adatmodellezés esetén olyan szoftver, amely segíti a modellezés teljes folyamatát

Tipikus funkciók

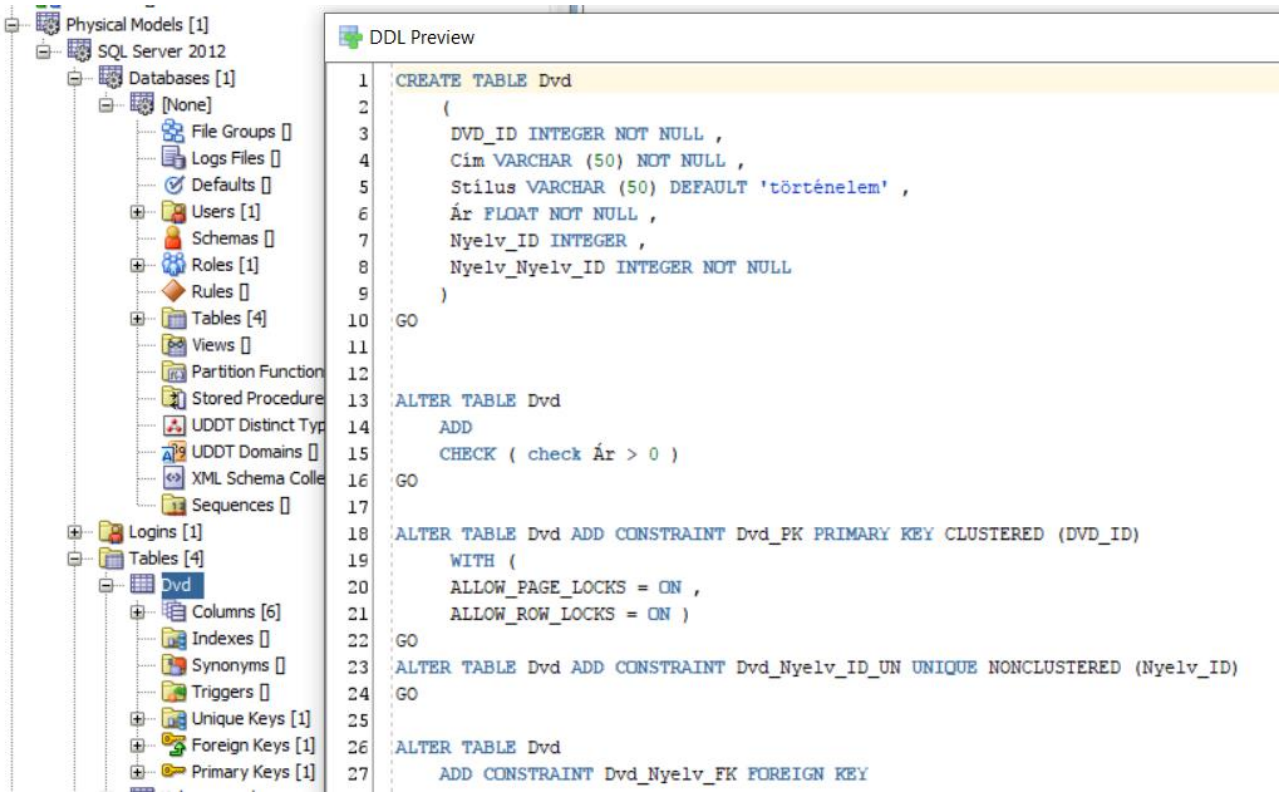
- ❑ Vizuális modellezés
- ❑ Adatstruktúrák tervezése
- ❑ Kódgenerálás
- ❑ Reverse engineering
- ❑ Dokumentáció
- ❑ Verziókezelés

# Pl: Oracle SQL Data Modeler

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




Relációs modell (SQL Server 2012)

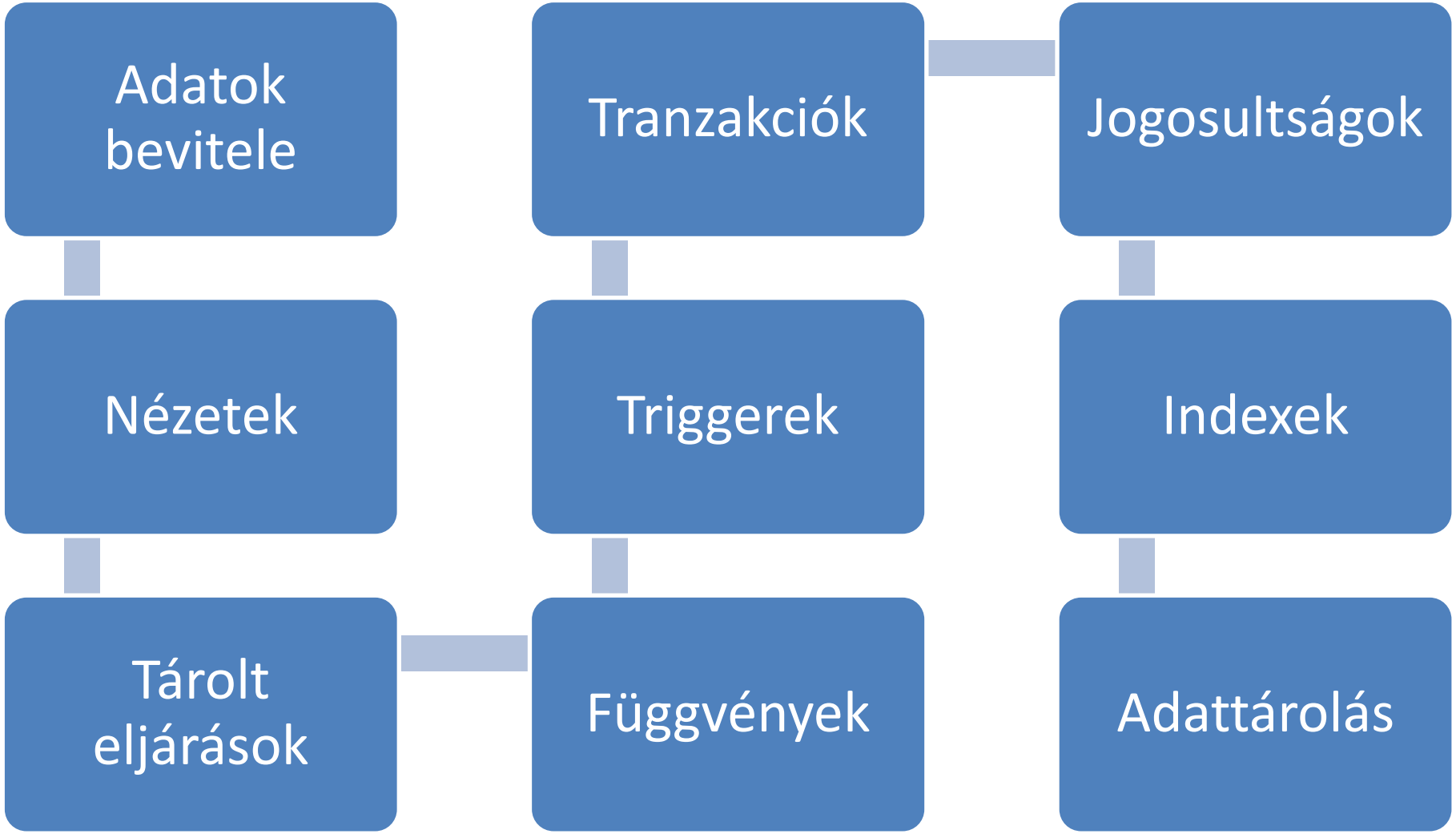



Fizikai modell + a Dvd tábla kódja

# 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

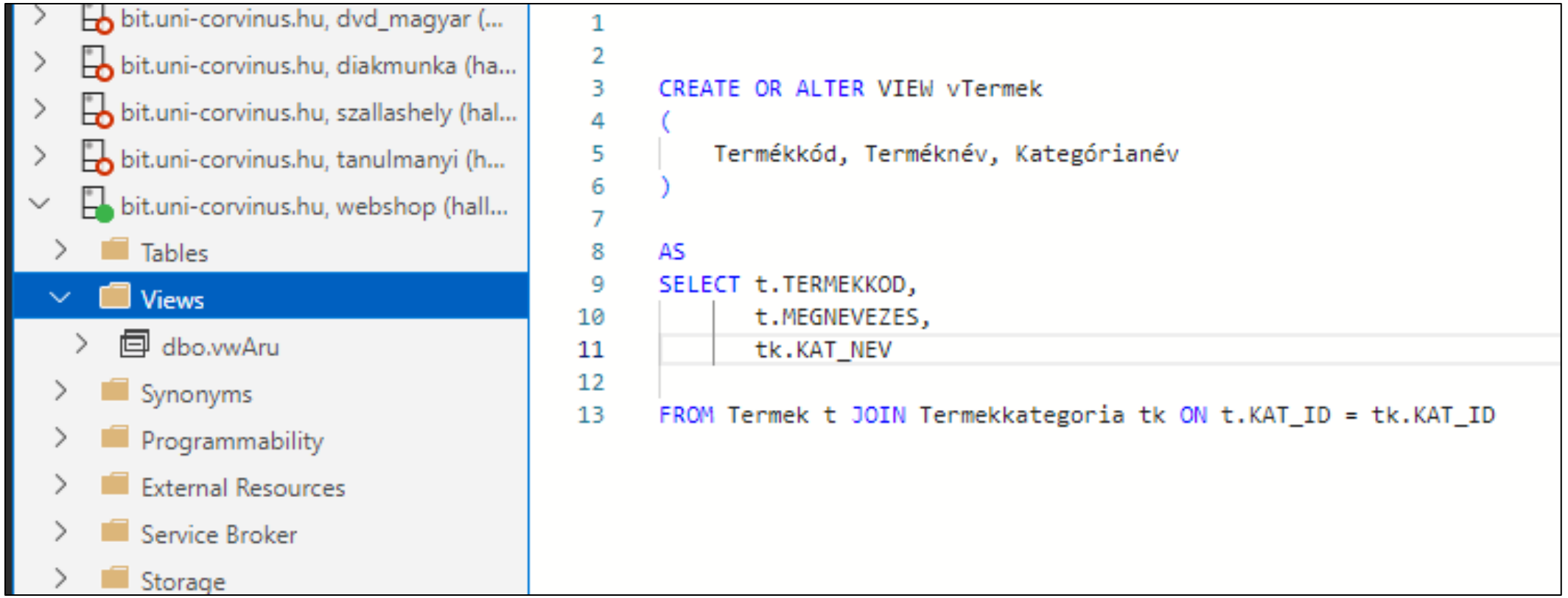
- Korlátozható az adatok elérése
- A bonyolultabb lekérdezések egyszerűbb formára hozhatók
- Az adatokat többféle nézőpontból szemlélhetjük
- Az adatfüggetlenség biztosítása

# 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

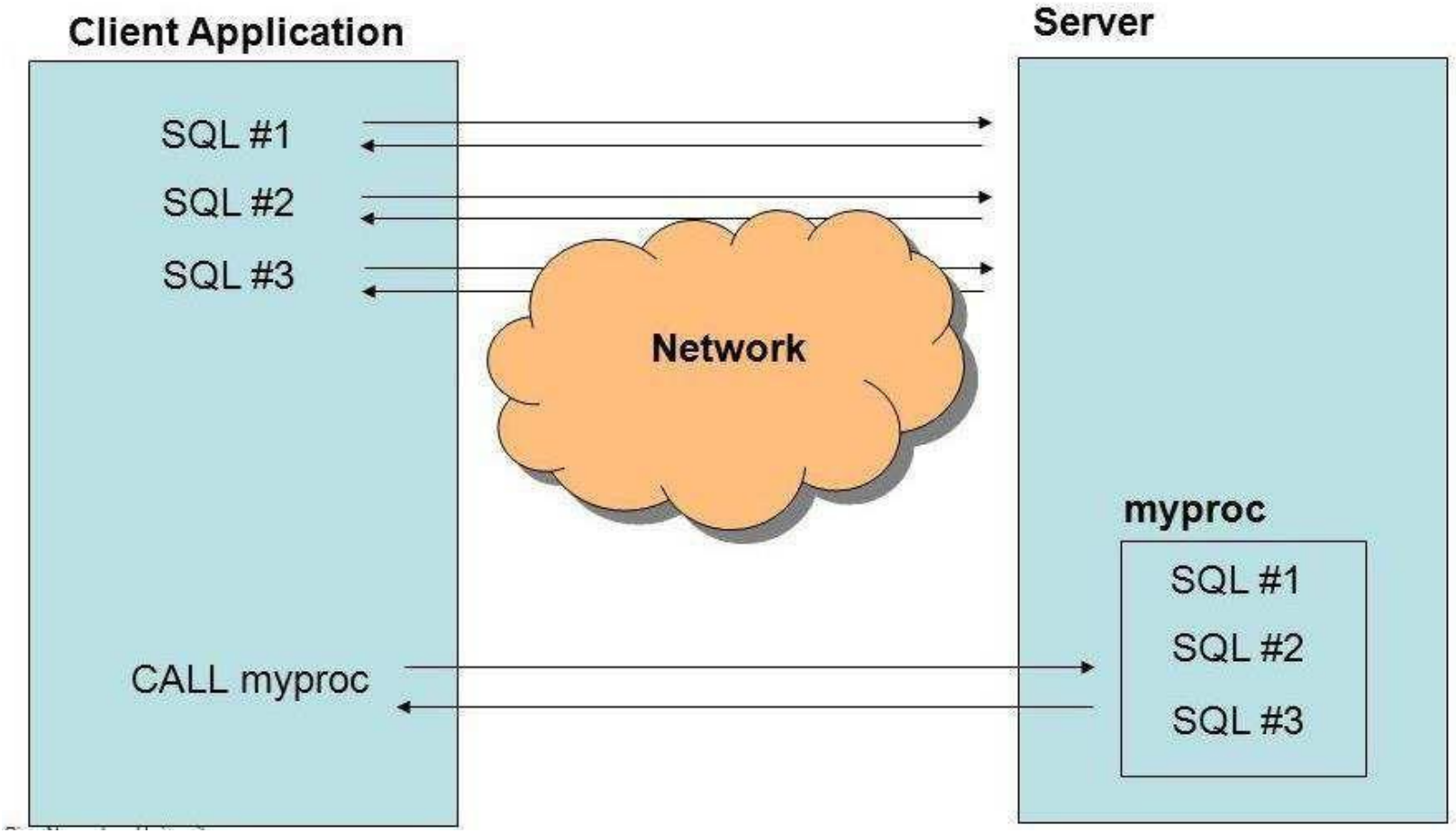
Mi a hátránya annak, ha az SQL kódot beépítjük a kliens alkalmazásba?

```
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.

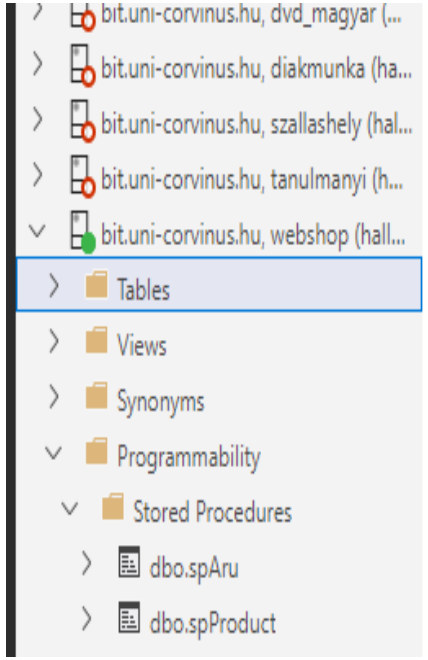
A tárolt eljárások főbb jellemzői

- 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<br>• Csökken a szerver-kliens üzenetek száma | • A kódok egy központi helyen találhatók<br>• A módosítás, tesztelés elkülönülhet a tárolt eljárást hívó alkalmazástól | • Használatukkal korlátozható a táblákhoz való hozzáférés<br>• A hozzáférés biztosítása így nem a tárolt eljárást hívó alkalmazás feladata | • Az üzleti logika elkülönül a tárolt eljárást hívó alkalmazástól<br>• 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

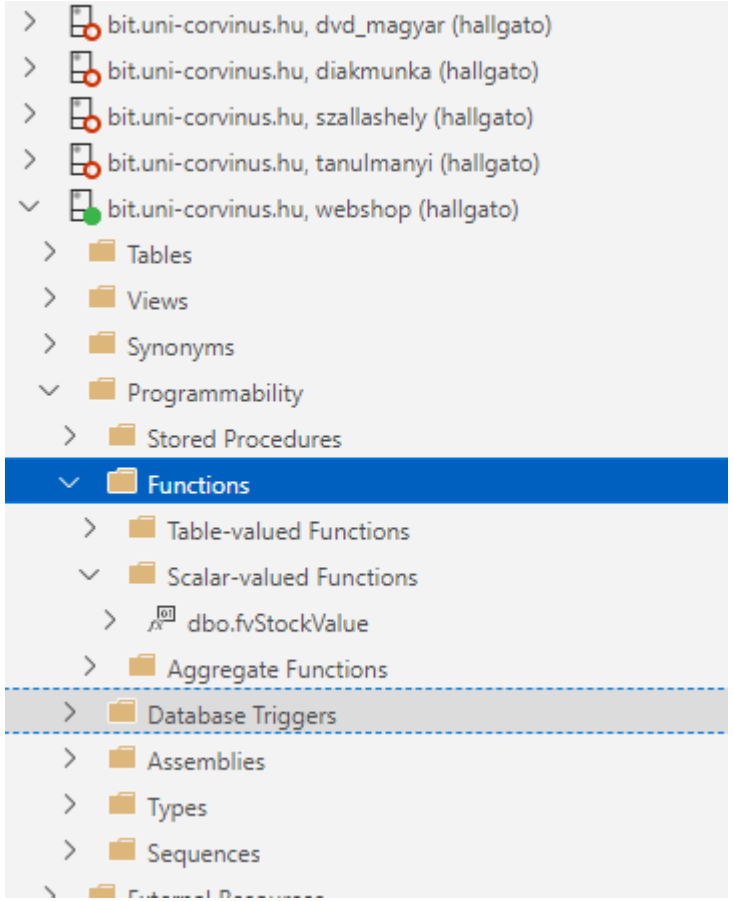
- Input paramétereket, SQL-utasításokat, valamint különböző algoritmikus szerkezeteket is tartalmazhatnak (elágazás, ciklus)
- Az adatbázis szerveren tárolódnak
- Futtatásuk jogosultságokhoz köthető
- Felhasználhatók SQL-utasításokban, pl: SELECT utasításban

# 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

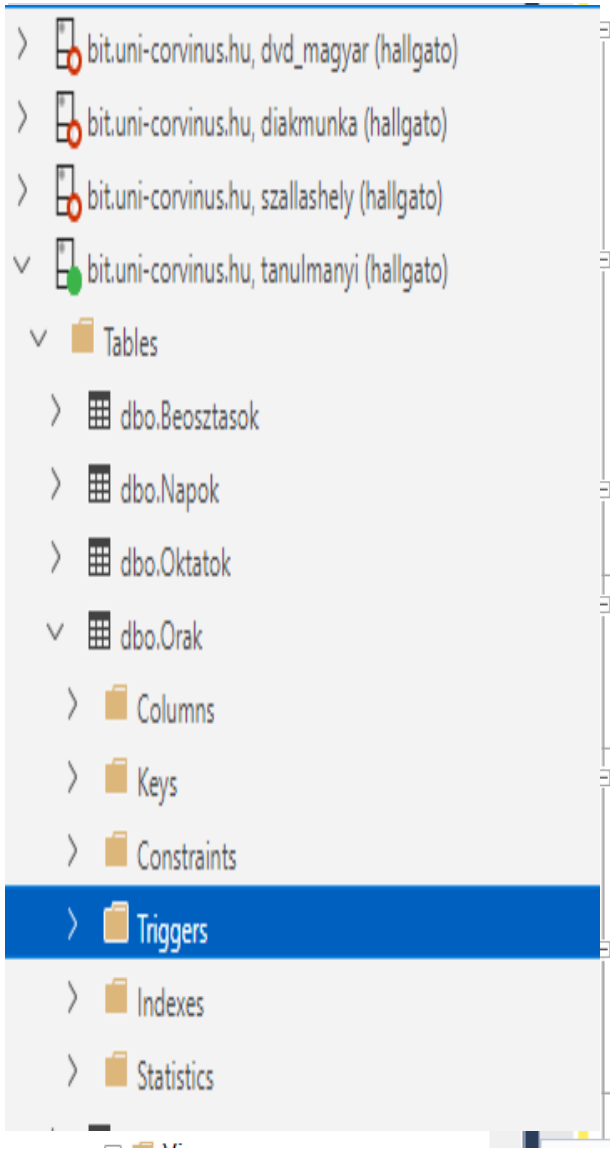
*Csak a DML triggerekkel foglalkozunk

# 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

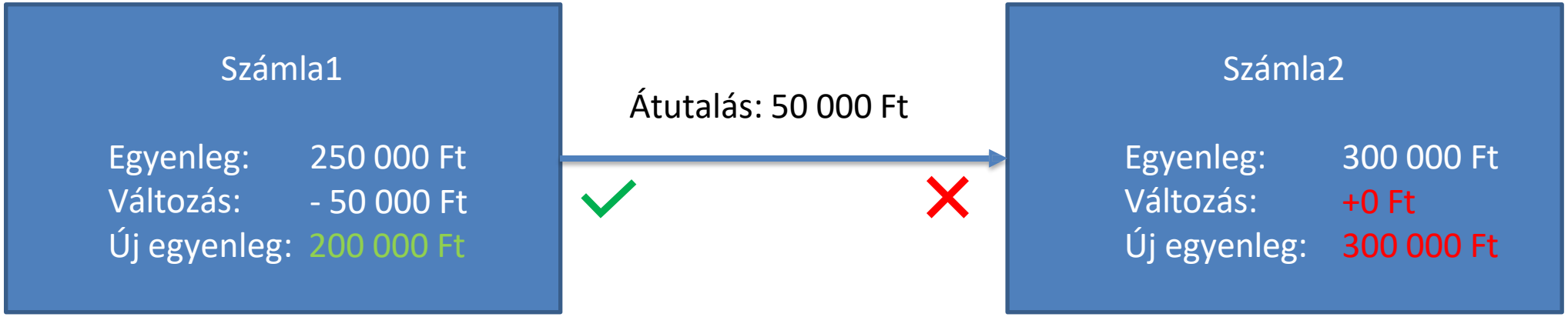

Az átutalás két fő lépése:

- 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

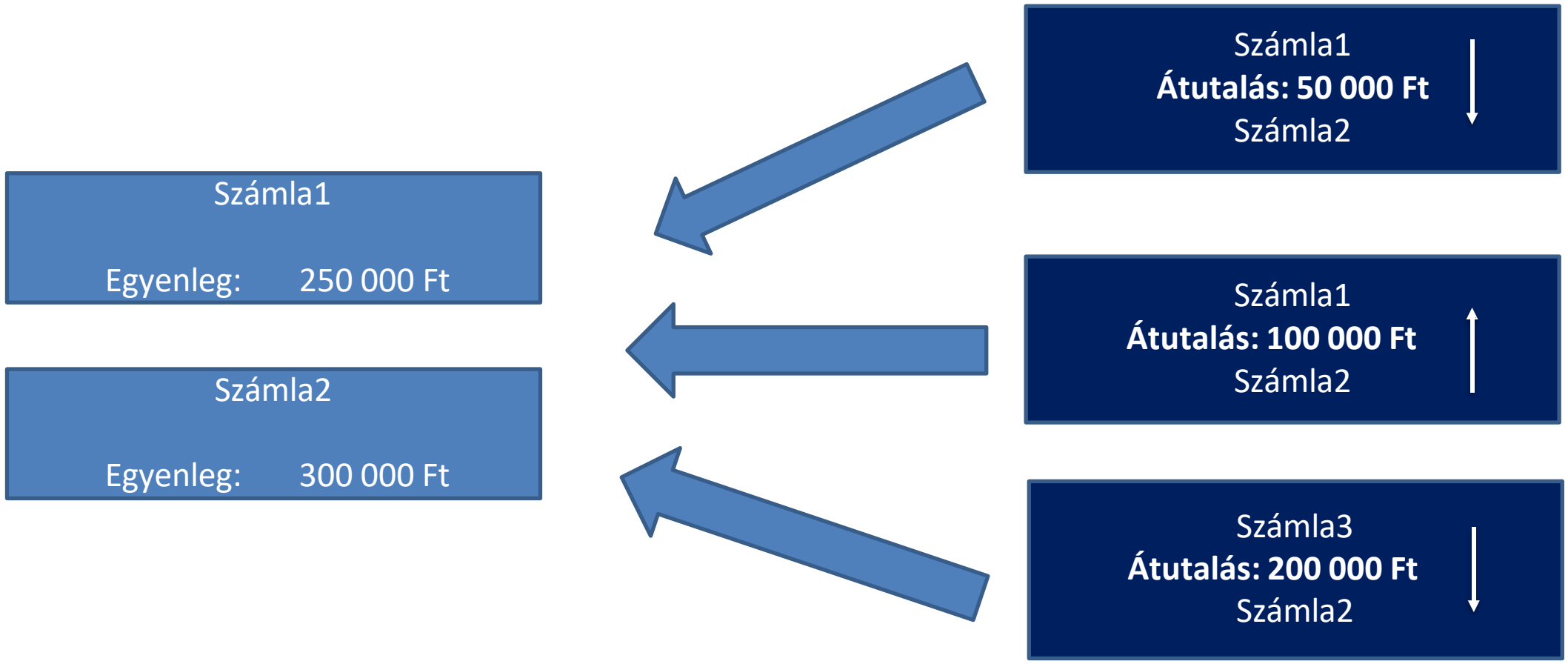
Számla2

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

Ha valahol megszakad a folyamat, akkor visszaáll az eredeti állapot

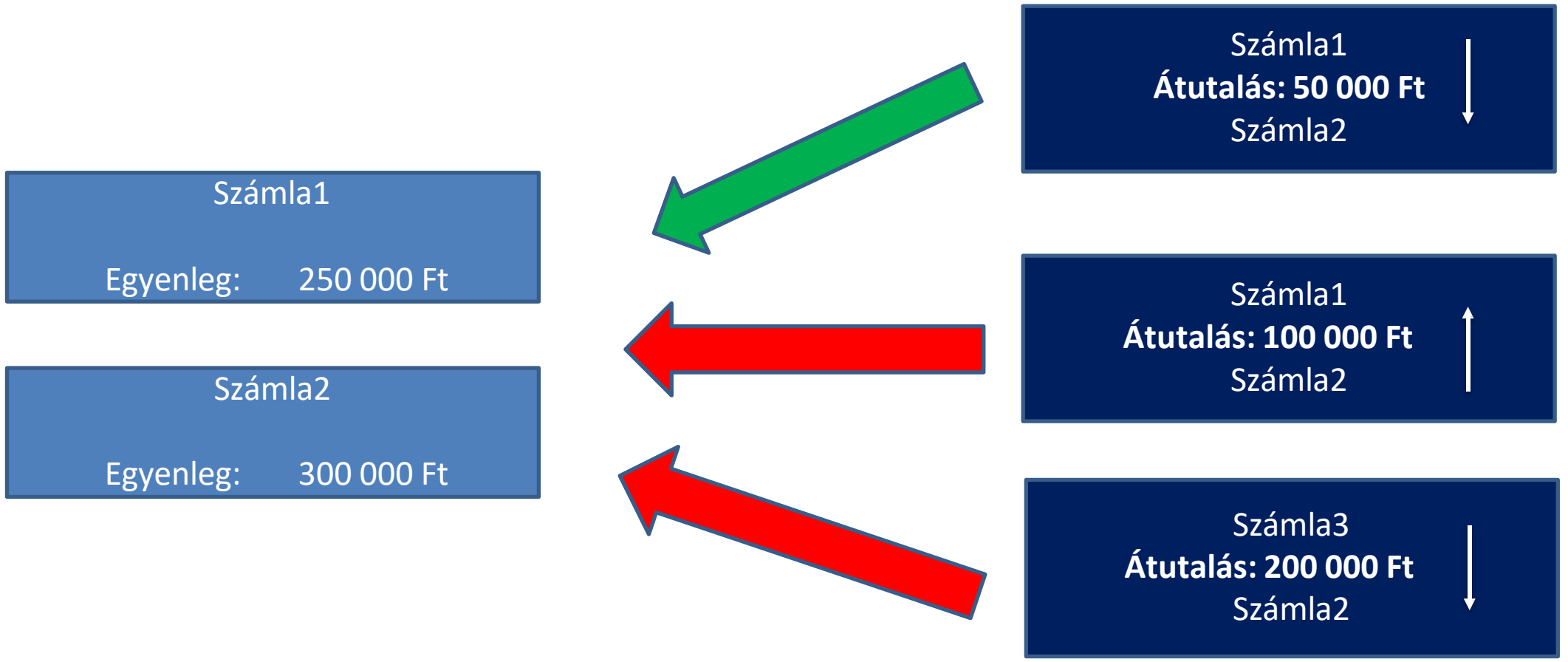
# Probléma2 – átutalás

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

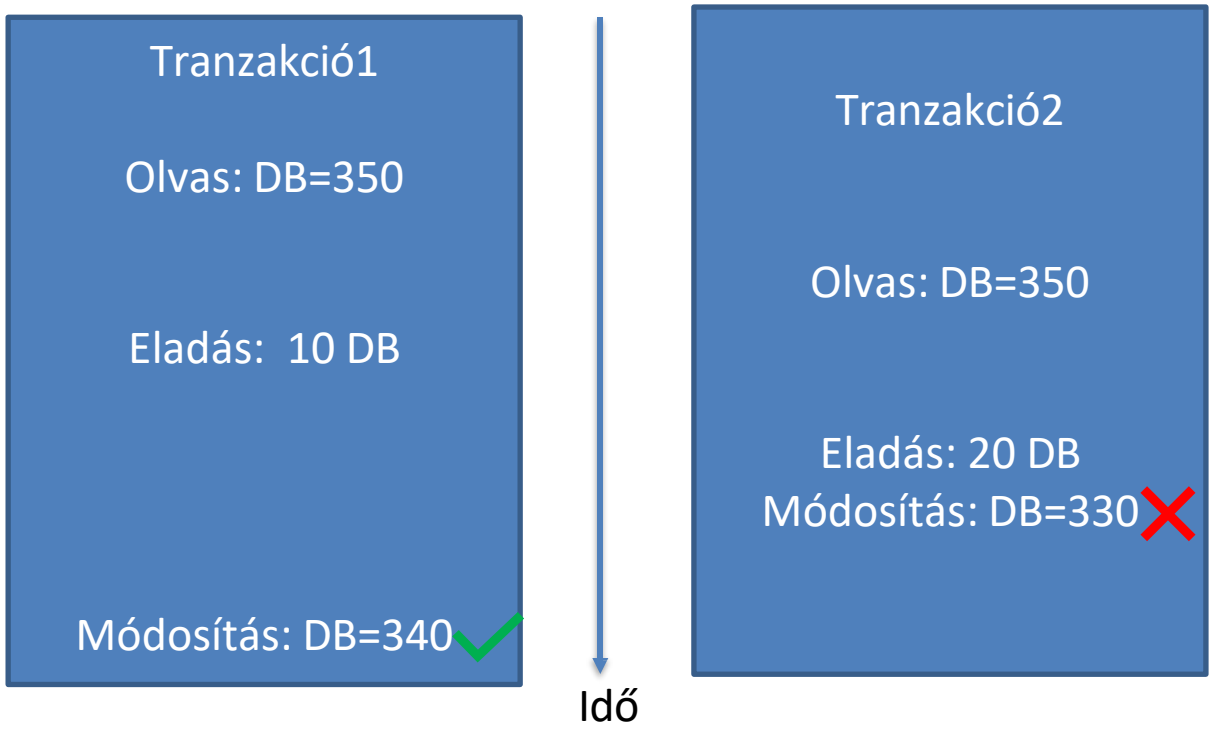
| Existing granted mode | IS | S | U | IX | SIX | X |
|---|---|---|---|---|---|---|
| Requested mode | | | | | | |
| Intent shared (IS) | Yes | Yes | Yes | Yes | Yes | No |
| Shared (S) | Yes | Yes | Yes | No | No | No |
| Update (U) | Yes | Yes | No | No | No | No |
| Intent exclusive (IX) | Yes | No | 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.

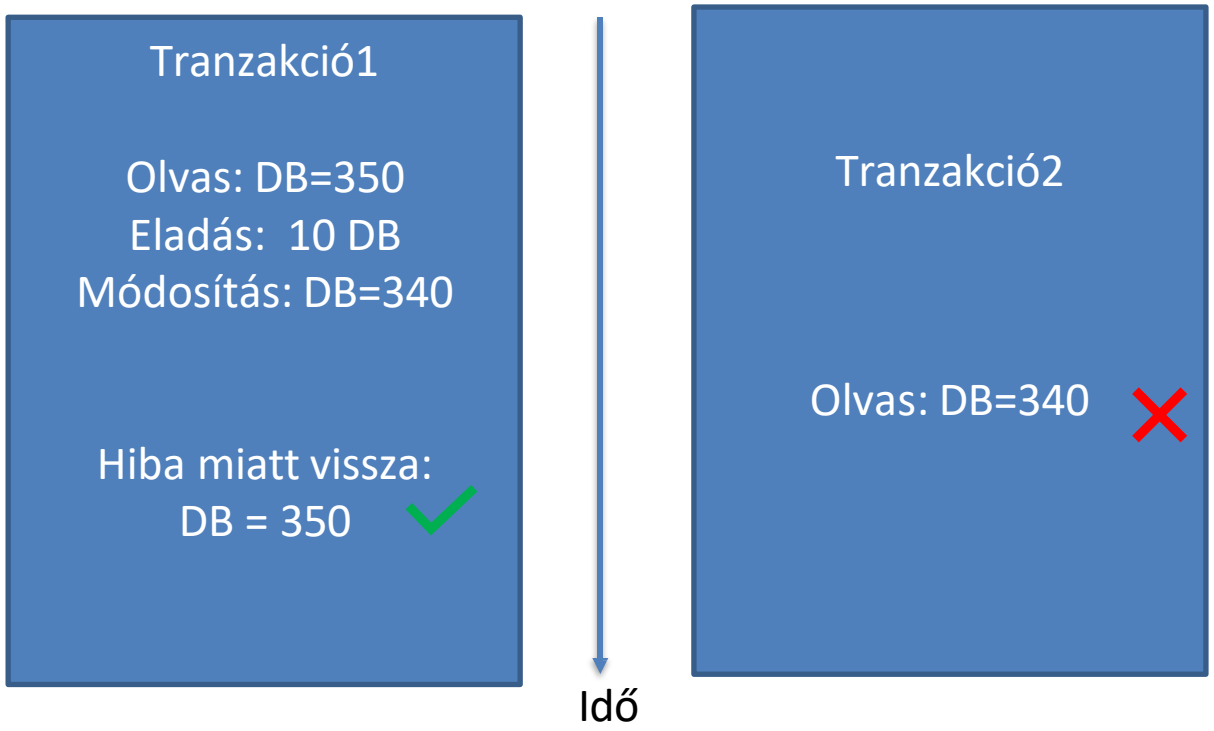
| KÉSZLET | | |
|---|---|---|
| ID | Termék | DB |
| 1 | Tégla | 350 |
| 2 | Cement 50 kg | 50 |



# Egyidejű tranzakciók kezelése (folyt.)

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

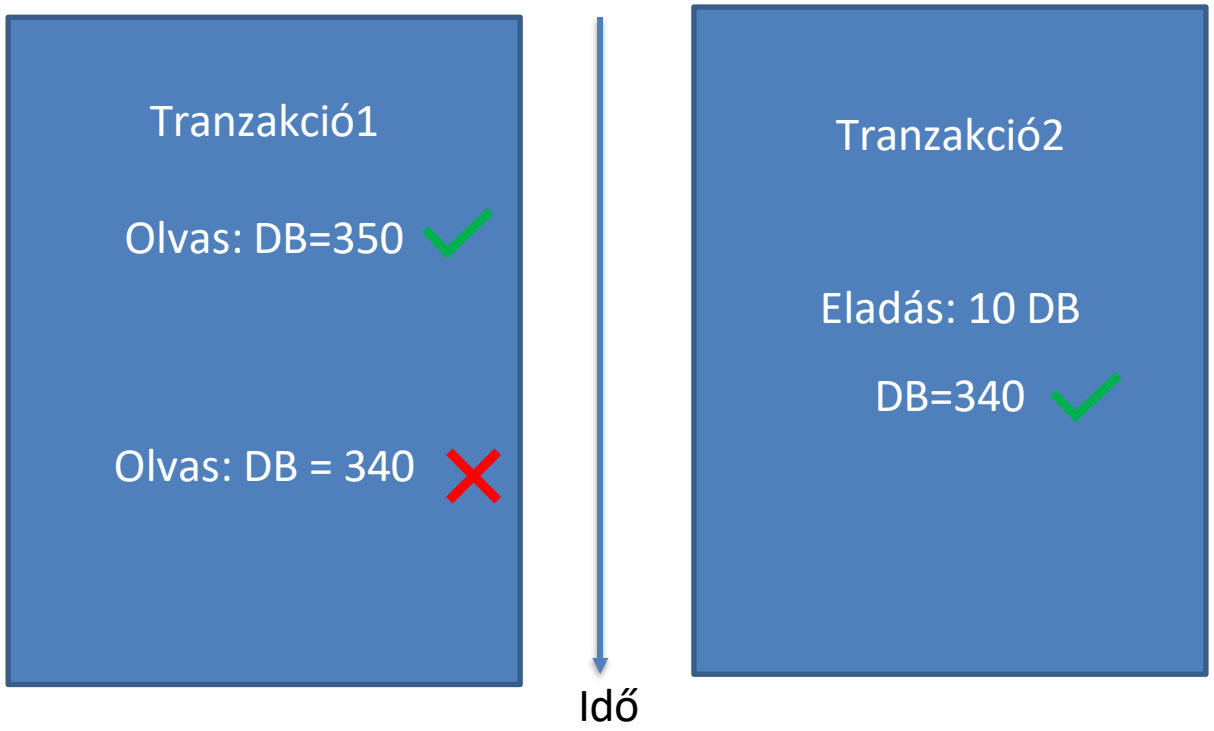
| KÉSZLET | | |
|---|---|---|
| ID | Termék | DB |
| 1 | Tégla | 350 |
| 2 | Cement 50 kg | 50 |



# 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.

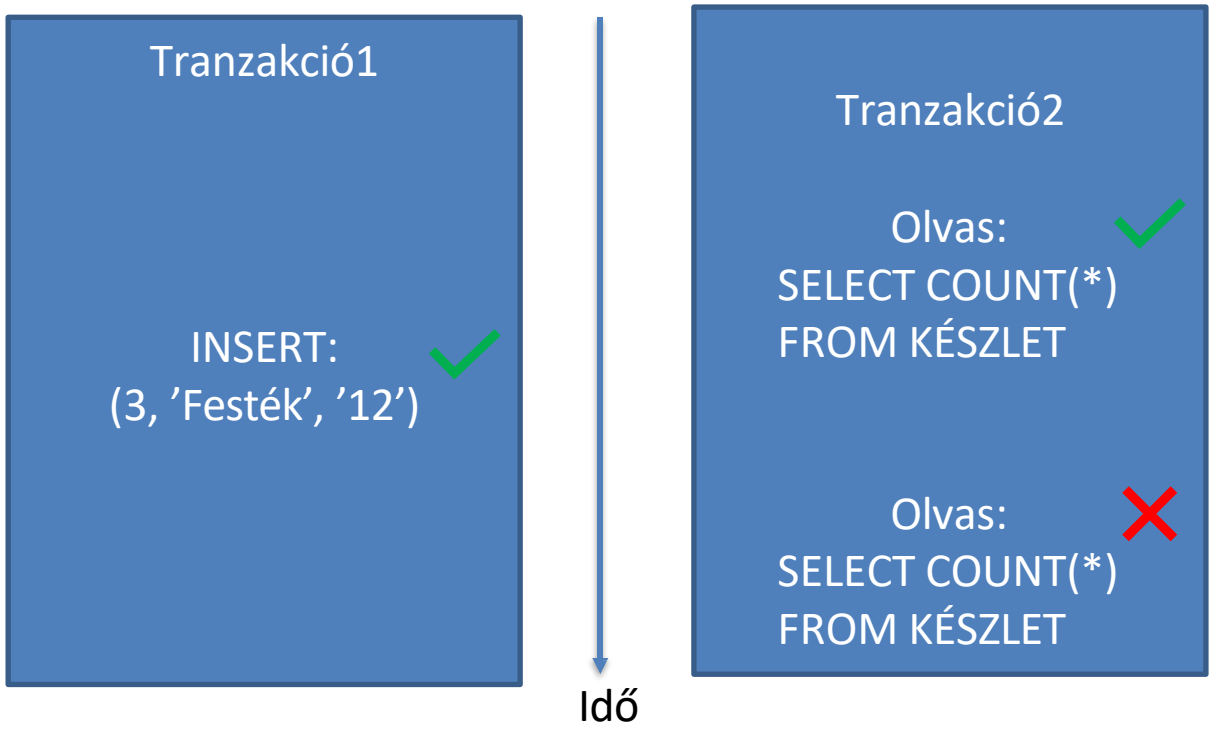
| KÉSZLET | | |
|---|---|---|
| ID | Termék | DB |
| 1 | Tégla | 350 |
| 2 | Cement 50 kg | 50 |



# 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;

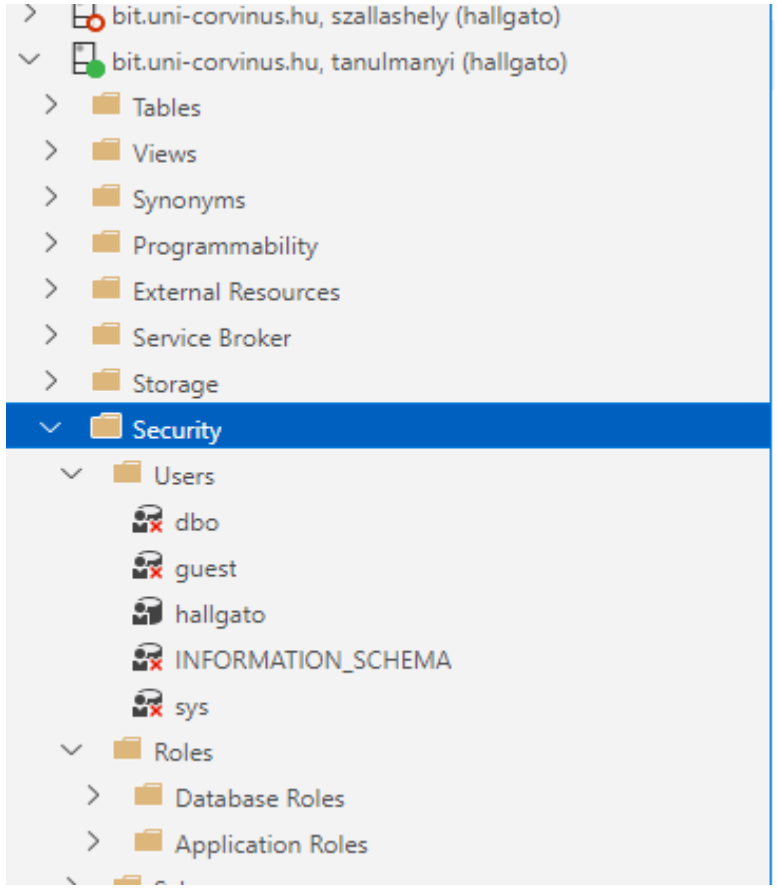
- AUTHORIZATION (engedély) lehet GRANT, REVOKE és DENY (jog adása, visszavonása és megtagadása)
- PERMISSION (a konkrét jogosultság), több, mint 200 féle, pl: SELECT, EXECUTE, UPDATE
- SECURABLE lehet szerver, szerver objektum, adatbázis, adatbázis objektum
- PRINCIPAL lehet LOGIN, felhasználó vagy szerepkör

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

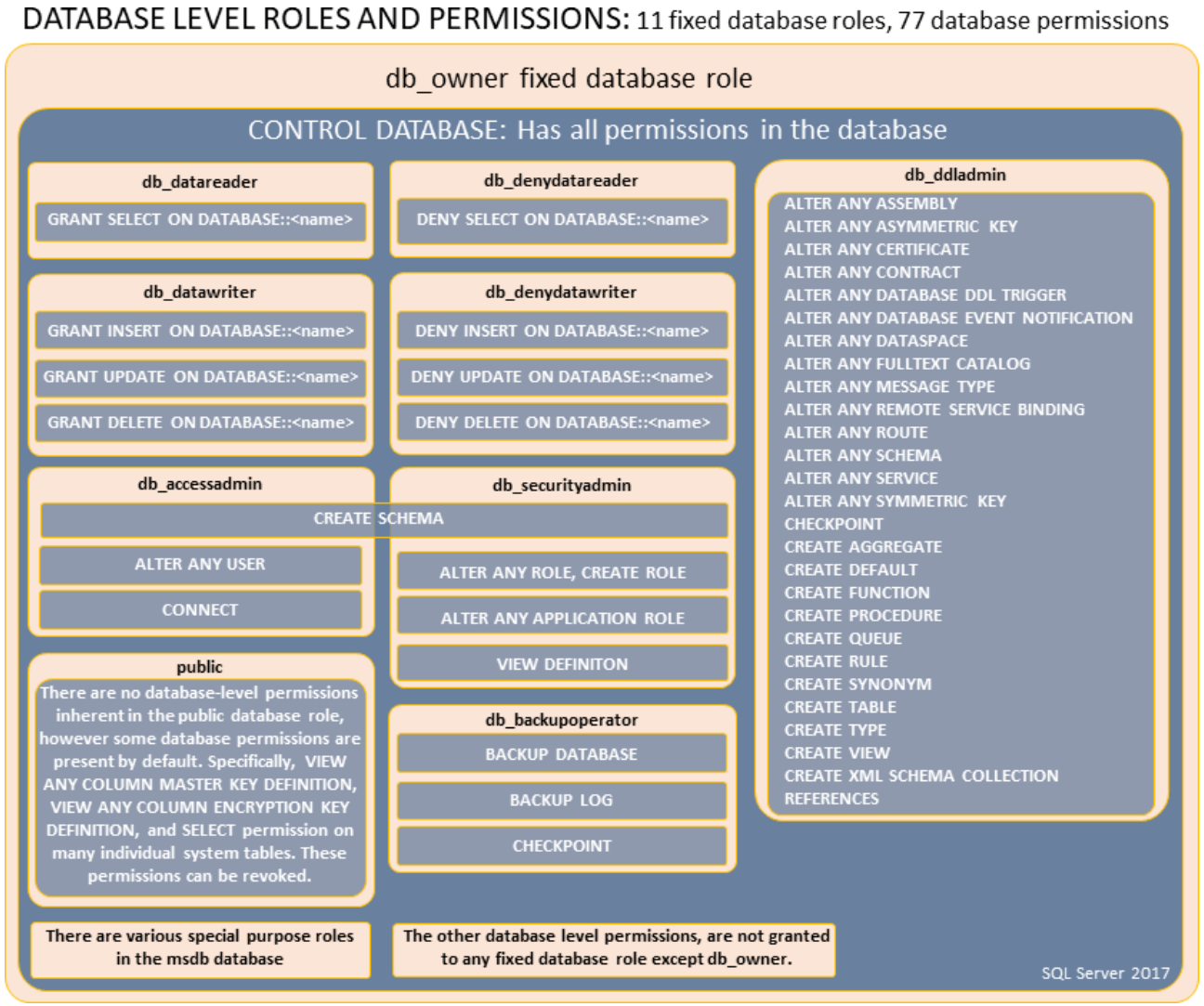
| Jogosultság | Rövid leírás |
|---|---|
| Sysadmin | Teljes joggal rendelkezik a szerveren |
| Serveradmin | Módosíthatja a szerver konfigurációt |
| Securityadmin | Szerver-szintű jogosultságokat kezelhet. Ha van hozzáférése adatbázisokhoz, akkor ott adatbázis-szintű jogokat is adhat vagy megtagadhat, 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

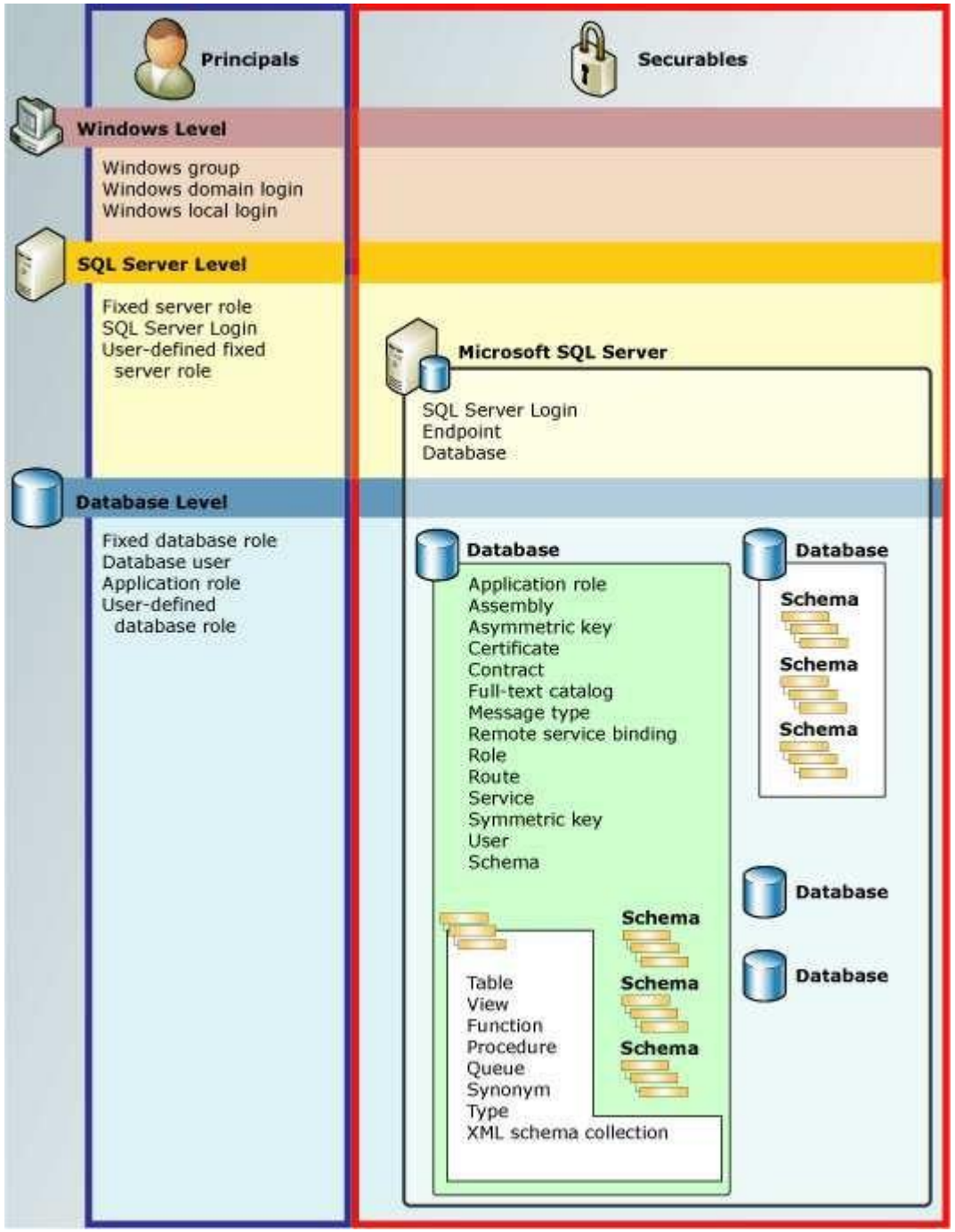


# Tábla jellegű objektumok jellemző jogosultságai (táblák, nézetek, tábla-értékű függvények)

| 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;
```

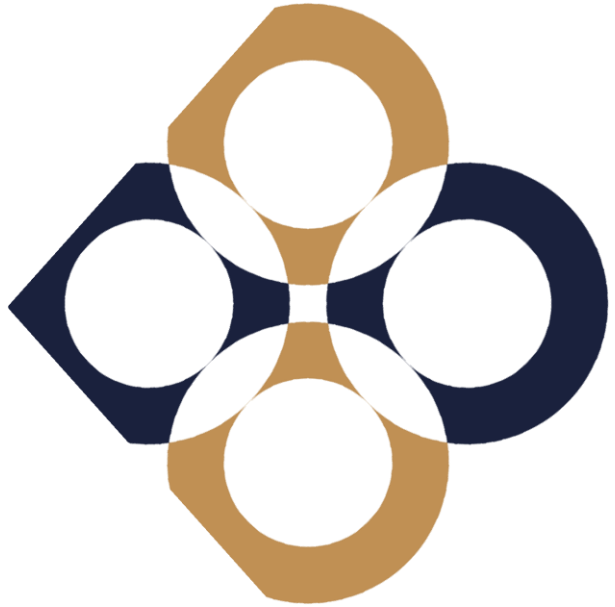
Results | Messages

| | 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 |

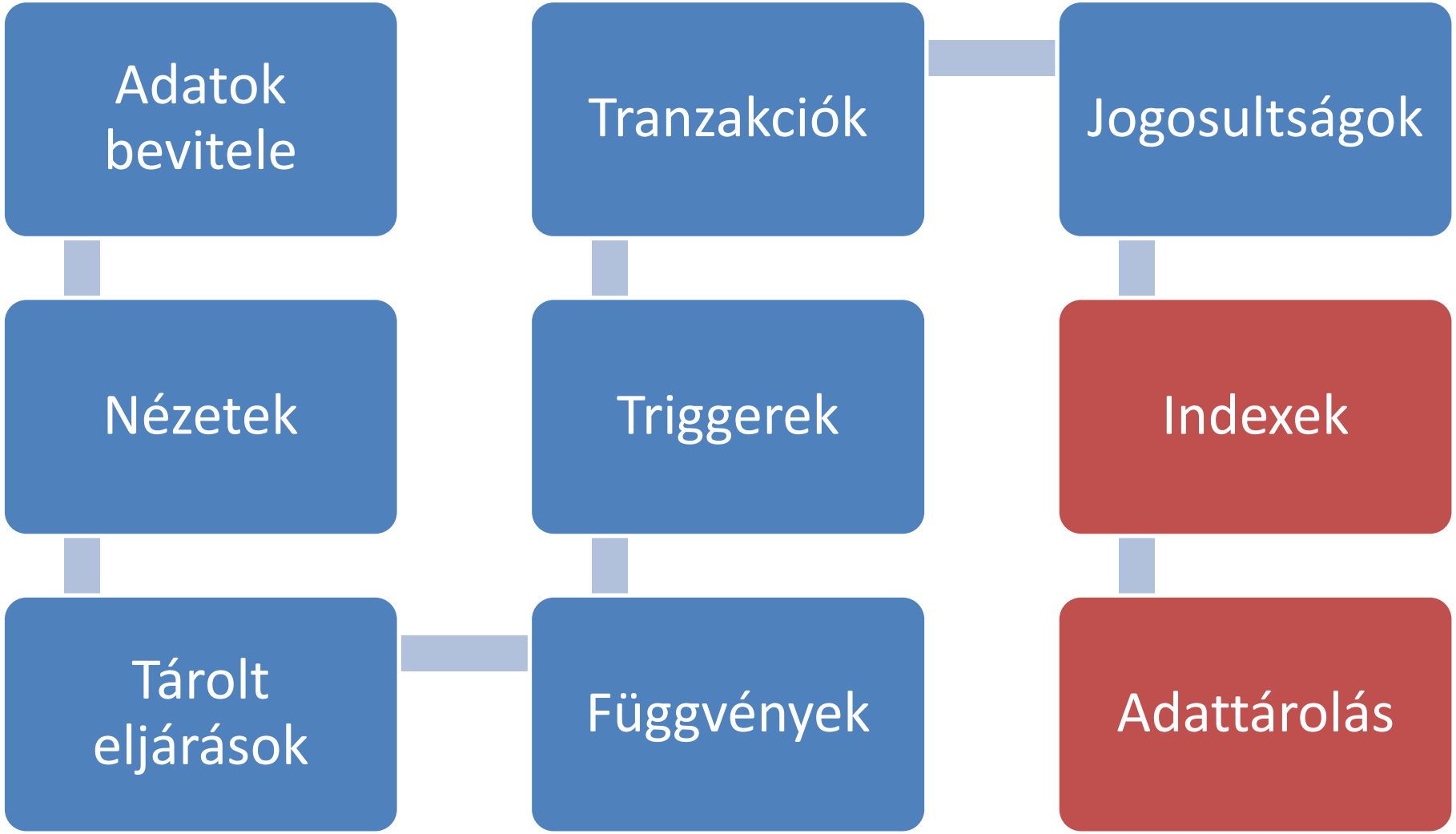
# Köszönöm a figyelmet!

BUDAPESTI CORVINUS EGYETEM

# Adatbázisok előadás 05



# Fejlesztési és konfigurálási feladatok

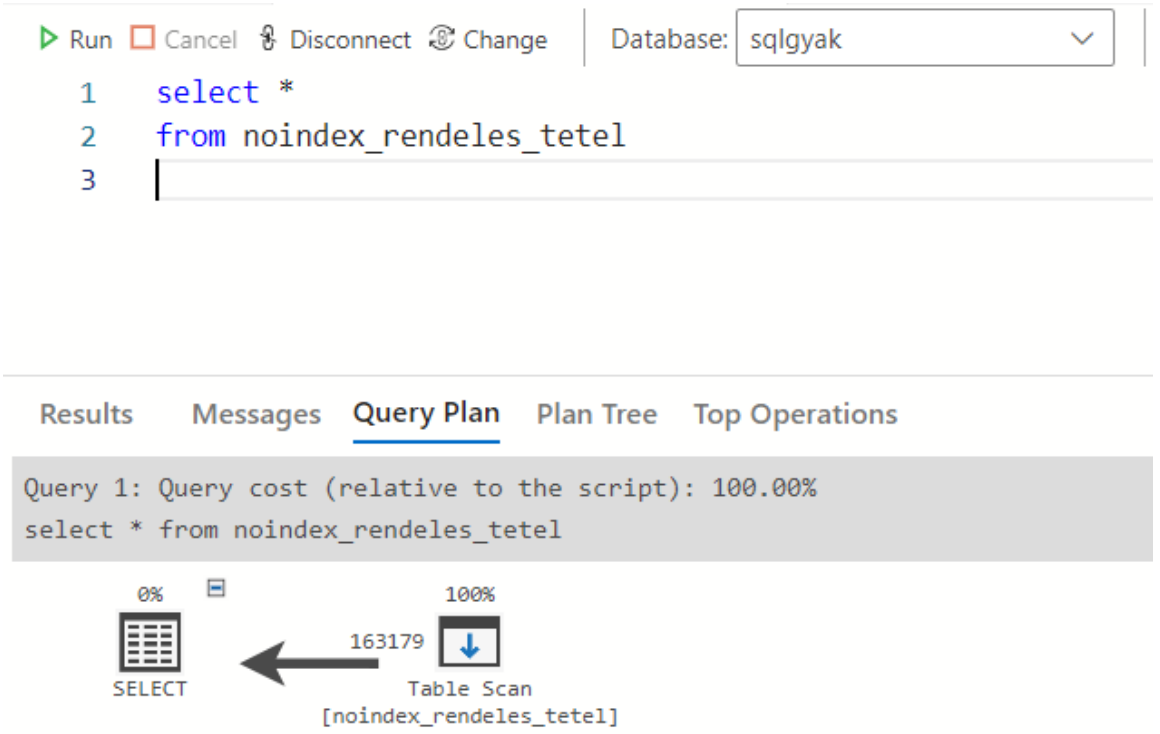


BUDAPESTI CORVINUS EGYETEM

# Indexek

# Hogyan lehet gyorsítani a lekérdezések sebességét?

Sok esetben az a probléma, hogy keresésnél akár az összes rekordot végig kell nézni (TABLE SCAN)




ÖTLET: Rendezzük sorrendbe az adatokat!

# Mi a baj a fizikai rendezéssel?

| | |
|---|---|
| Sok adatot kell mozgatni | Vagy növekvő, vagy csökkenő |
| Csak egyféle szempont | DML utasításoknál újra kell rendezni |

ÖTLET2: Rendezzük logikailag sorrendbe az adatokat!

# Mit jelent a logikai rendezés?

Csak a rendezés alapjául szolgáló mezőt (vagy kifejezést) és a rekordok azonosítóját (mutató, memóriacím) tároljuk

- Az adatok eredeti tárolási sorrendje nem változik
- Egy táblához több logikai rendezést is létrehozhatunk
- A logikai rendezést indexelésnek is nevezik

# Indexek

Az index a táblához vagy nézethez rendelt olyan speciális adatstruktúra, amely felgyorsítja a lekérdezések sebességét.

| INDEX (Név szerint) | |
|---|---|
| Név | ID |
| Bódi István | D02 |
| Fehér Katalin | D04 |
| Kiss Béla | D01 |
| Nagy Ilona | D03 |

| DOLGOZÓ | | | |
|---|---|---|---|
| ID | Név | Életkor | … |
| D01 | Kiss Béla | 22 | |
| D02 | Bódi István | 18 | |
| D03 | Nagy Ilona | 32 | |
| D04 | Fehér Katalin | 18 | |

# Indexek csoportosítása

## Egyedi ← → Duplikált

- Egy index érték csak egyszer fordulhat-e elő?

## Sűrű ← → Ritka

- Minden adatrekordhoz készül index bejegyzés?

## Egyszerű ← → Összetett

- Egy vagy több mezőre épül-e?

## Növekvő ← → Csökkenő

- Milyen irányú a rendezés?

# Sűrű vs. Ritka index


Az indexmutató egy rekordra mutat.

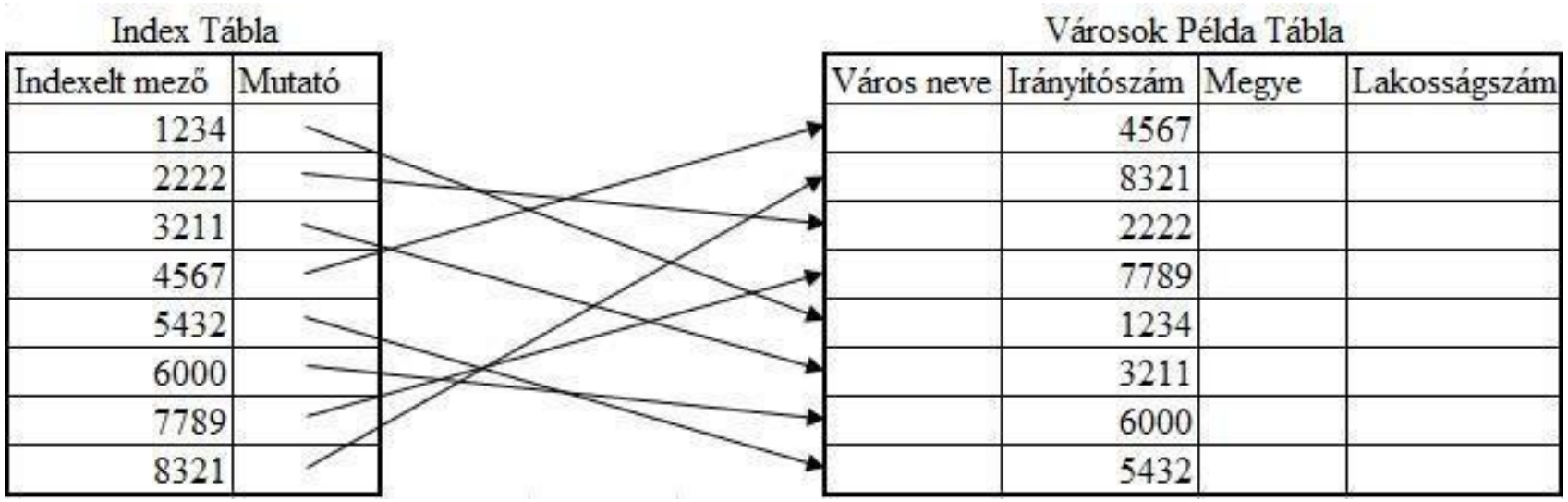

Az indexmutató egy blokkra mutat.
A blokkon belül a keresés szekvenciális

# Index adatstruktúrák

- Egyszintes indexek
- Többszintes indexek
- B-fák
- Hash-alapú indexek
- Bitmap indexek

# Egyszintes indexek

Két mezőből álló indextábla, amely az indexelt mező alapján sorba van rendezve. A mutató a rekord fizikai helyére mutat.



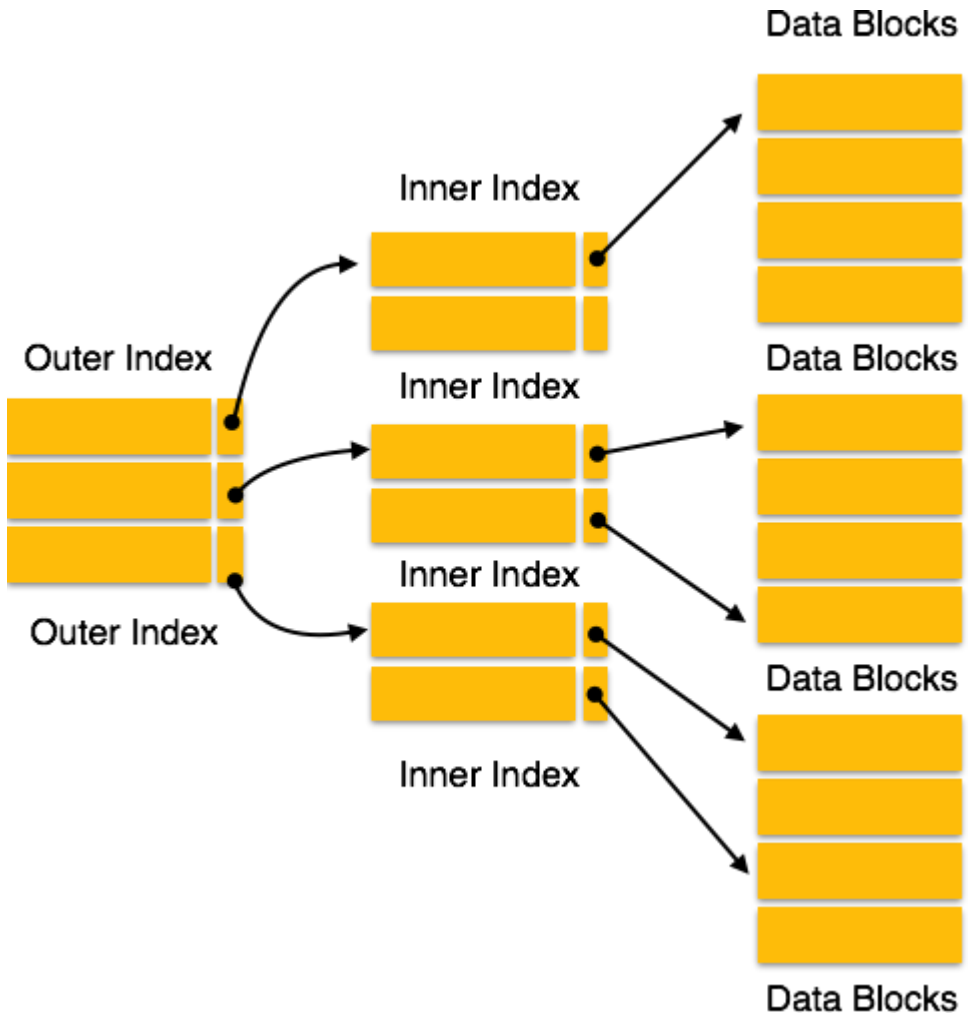
Index Tábla

| Indexelt mező | Mutató |
|---|---|
| 1234 | |
| 2222 | |
| 3211 | |
| 4567 | |
| 5432 | |
| 6000 | |
| 7789 | |
| 8321 | |

Városok Példa Tábla

| Város neve | Irányítószám | Megye | Lakosságszám |
|---|---|---|---|
| | 4567 | | |
| | 8321 | | |
| | 2222 | | |
| | 7789 | | |
| | 1234 | | |
| | 3211 | | |
| | 6000 | | |
| | 5432 | | |

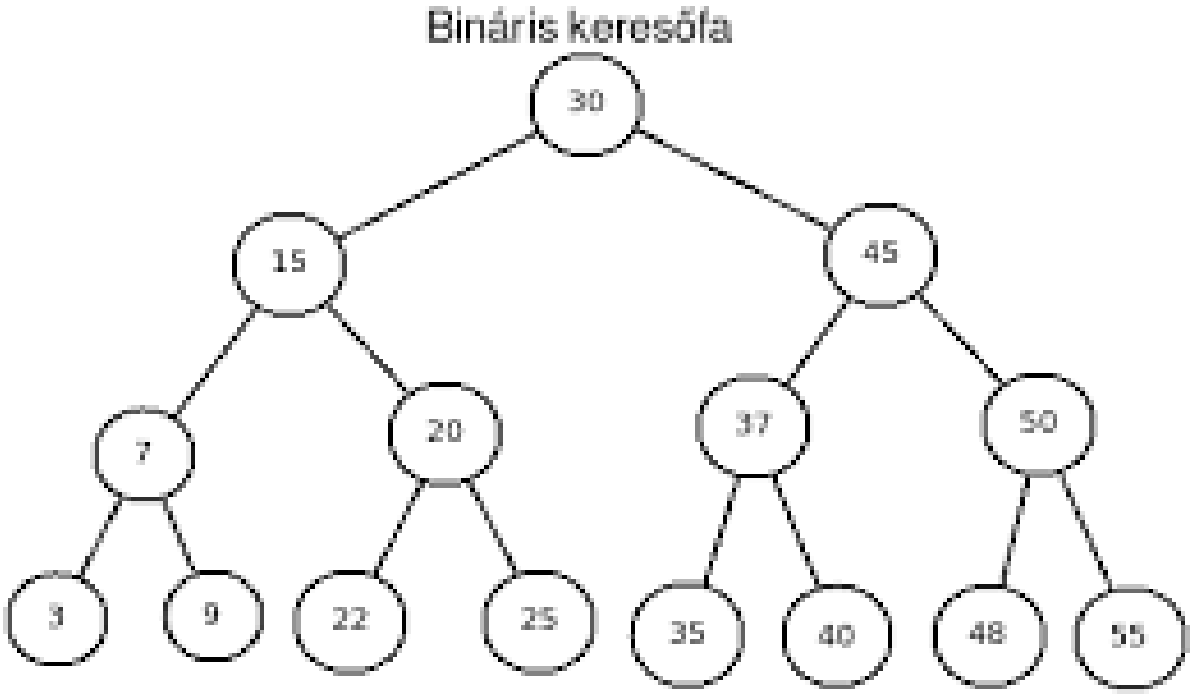
# Többszintes indexek




Az indexekhez is indexet készítünk

- Az index így kisebb részekből áll
- Hasznos, ha az egyszintes index nem fér el a memóriában
- Kevesebb blokk olvasás szükséges az adat megtalálásához

# Keresőfák




Bármely csomópontból kiindulva a csomópont bal részfájában csak a csomópontban elhelyezettnél kisebb, a jobb részfájában pedig csak nagyobb értékek szerepelnek

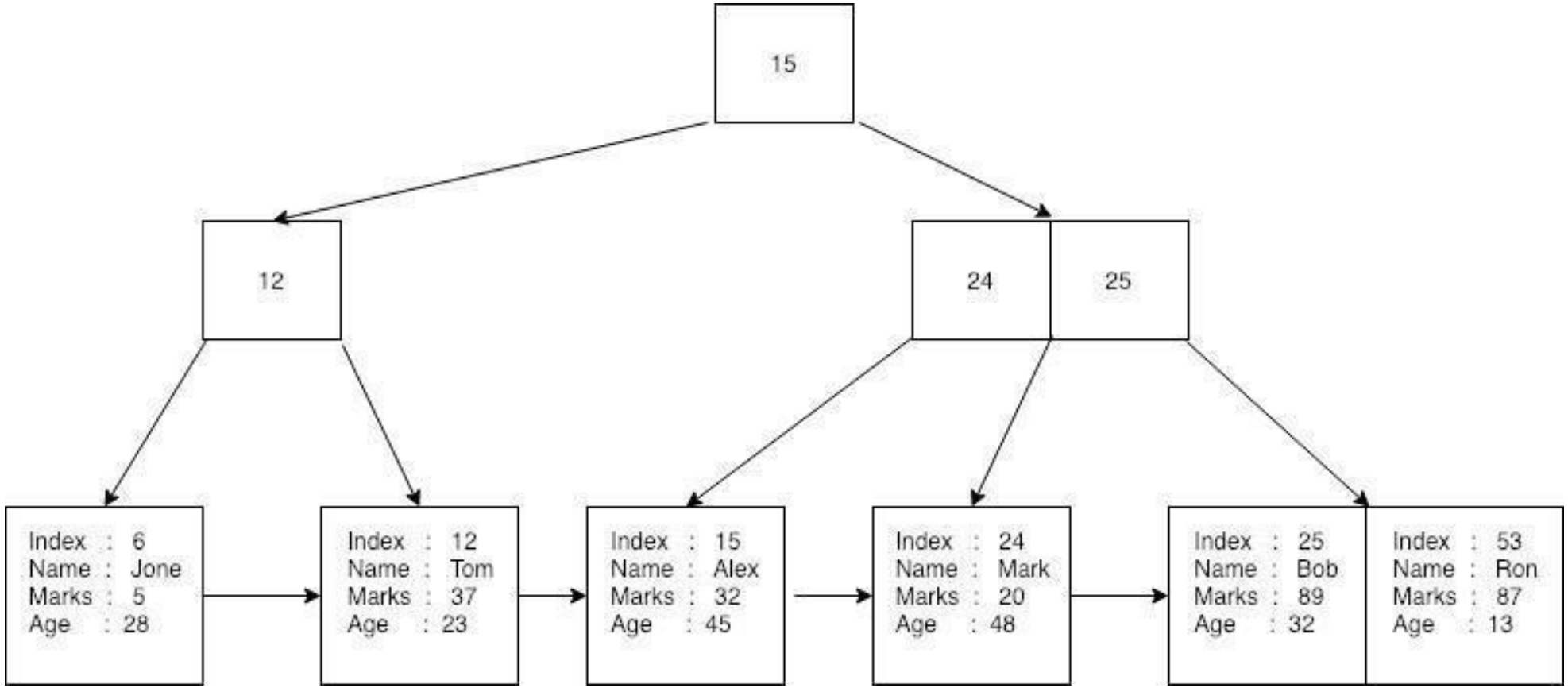
BUDAPESTI CORVINUS EGYETEM

# B-fák tulajdonságai

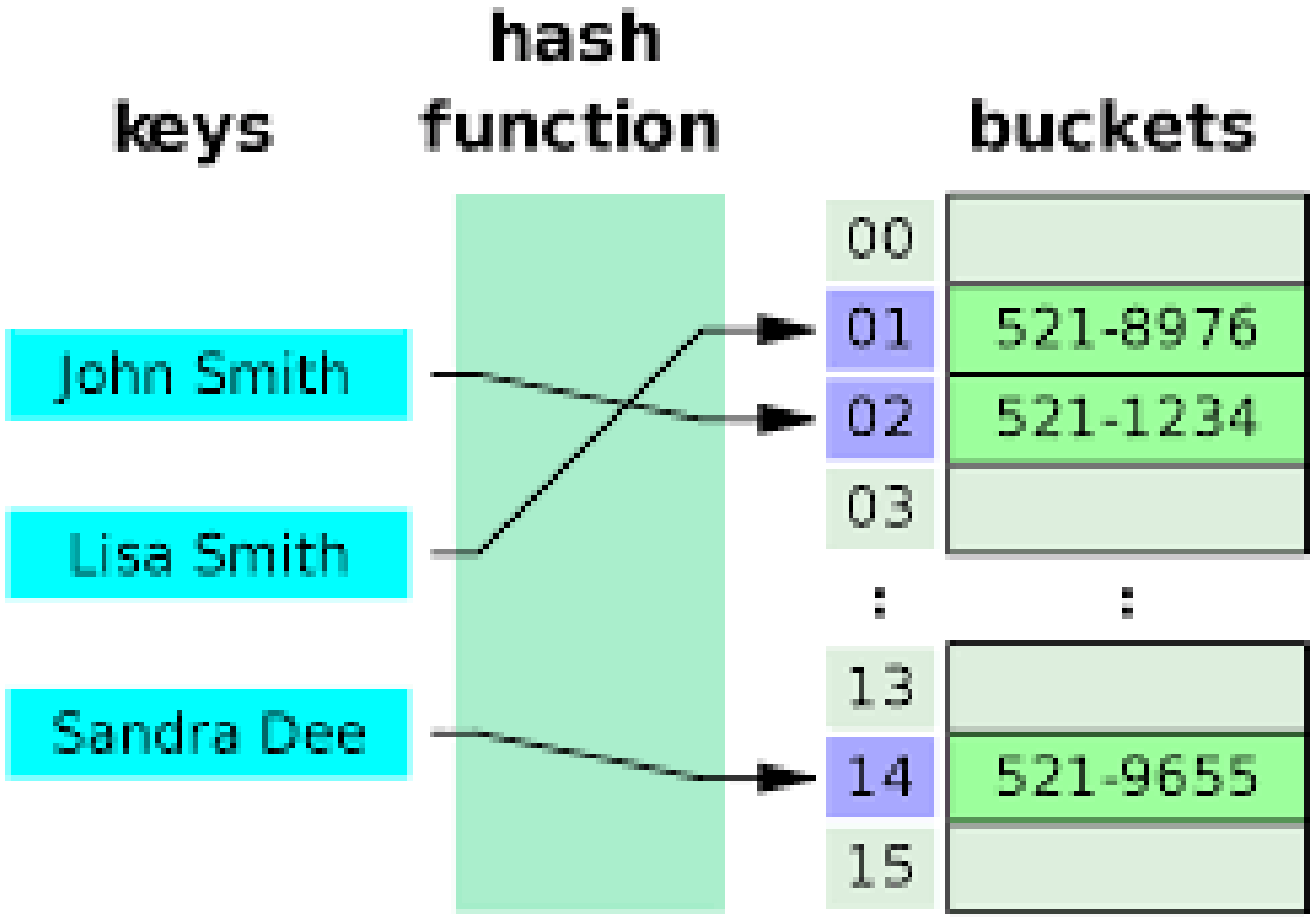
- A gyökértől a levelekig vezető utak hossza egyforma
- Az indexek a B-fa csomópontjaiban helyezkednek el
- Az adatok helyét jelző mutató csak a levelekben található
- A struktúra lehetővé teszi a soros és a random elérést is

BUDAPESTI CORVINUS EGYETEM

# B-fák (kiegyensúlyozott keresőfák)

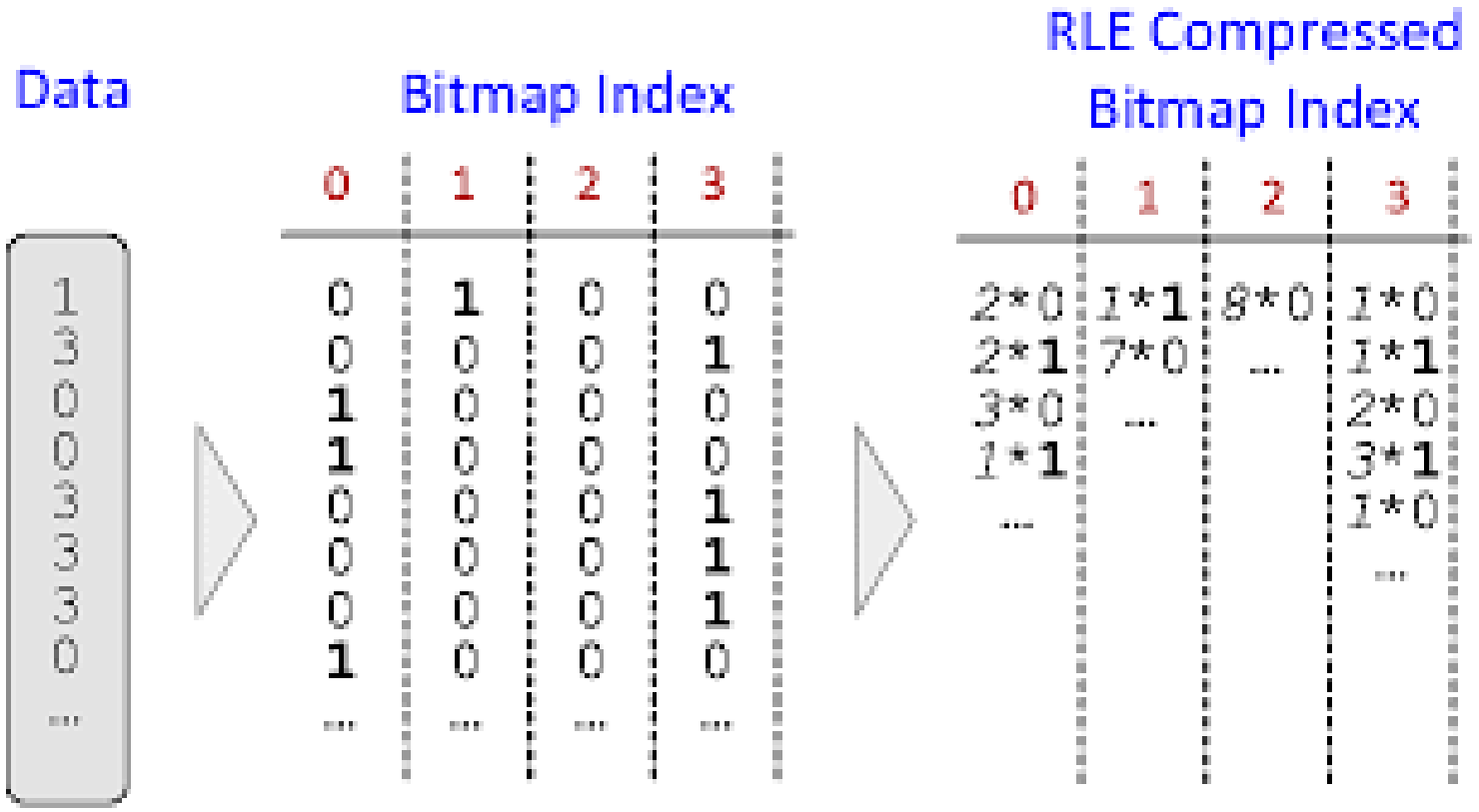


# Hash-alapú indexek




- Az adatok csoportokba vannak rendezve
- A hash függvény adja meg, hogy melyik csoportban van az adat

# Bitmap indexek




- Olyan oszlopokra alkalmazzuk, ahol kevés az egyedi érték
- A bitmap index tömöríthető is

# Fontosabb T-SQL Index típusok

- Clustered
- Non-clustered
- Columnstore

Az indexek létrejöhetnek automatikusan vagy manuálisan (CREATE INDEX)

# Clustered index

Az adatokat az index kulcsnak megfelelő sorrendbe rendezi és tárolja.

- A clustered index B-fa struktúrát használ
- Egy tábla esetén csak egy clustered index hozható létre
- Alapértelmezés szerint az elsődleges kulcs definiálásakor automatikusan létrejön

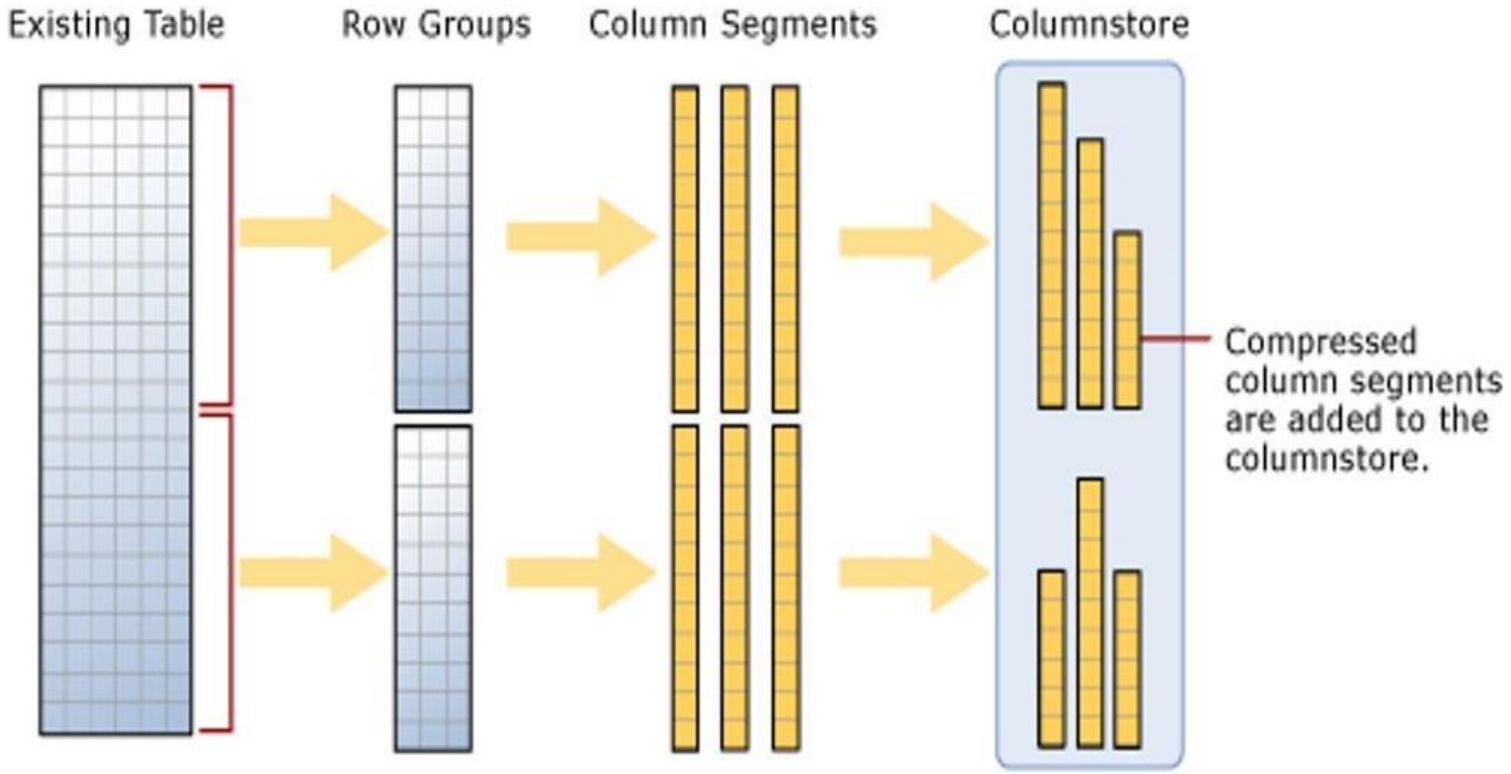
# Non-clustered index

A non-clustered index kulcs-mutató érték párokat tárol.

- Az index sorai a kulcs értékeknek megfelelő sorrendben vannak tárolva
- Az adatok tárolási sorrendje ettől eltérő
- Egy tábla esetén több non-clustered index is létrehozható

# Columnstore index

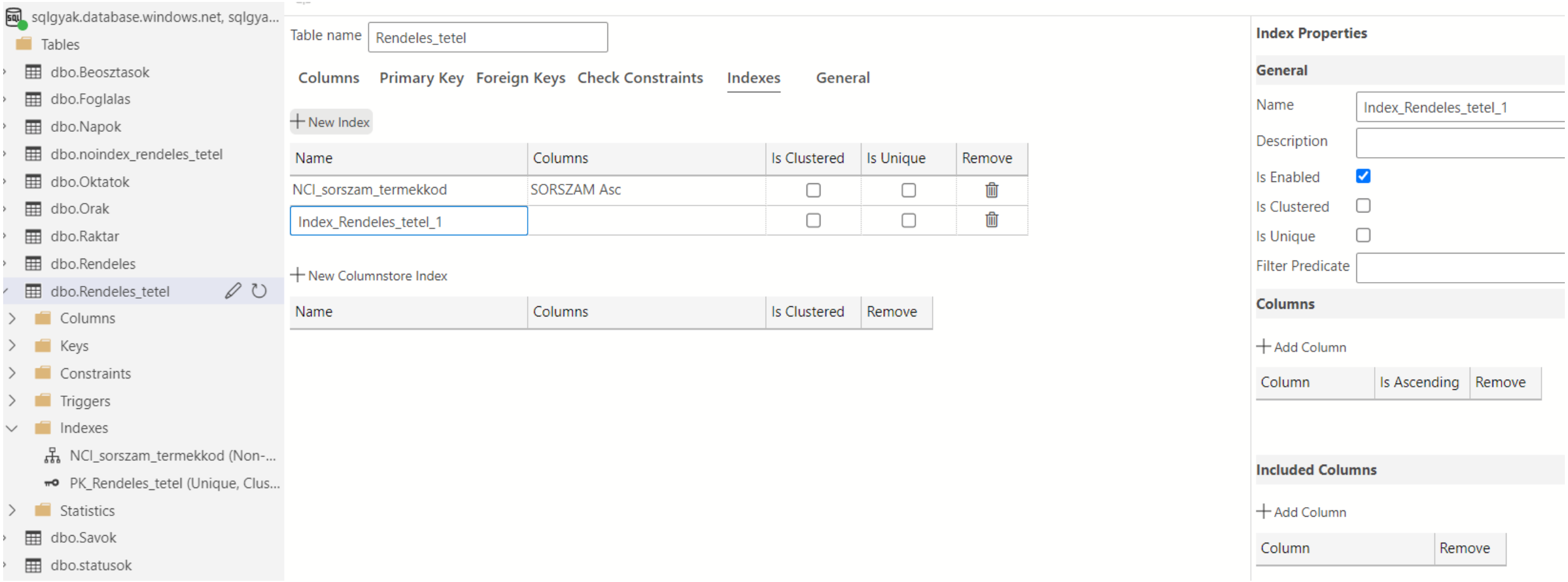
A columnstore index jellemzője az oszlop-alapú tárolás és lekérdezés végrehajtás




- A columnstore index lehet
  - Clustered
  - Non-clustered
- Egy táblához csak egy columnstore index készíthető

Adattárházakból való lekérdezéseknél kiemelten fontos!

# Indexek – Azure Data Studio



# Melyik index legyen használva?

HINT- Lekérdezési tipp

Használata:

SELECT ….

FROM…

…

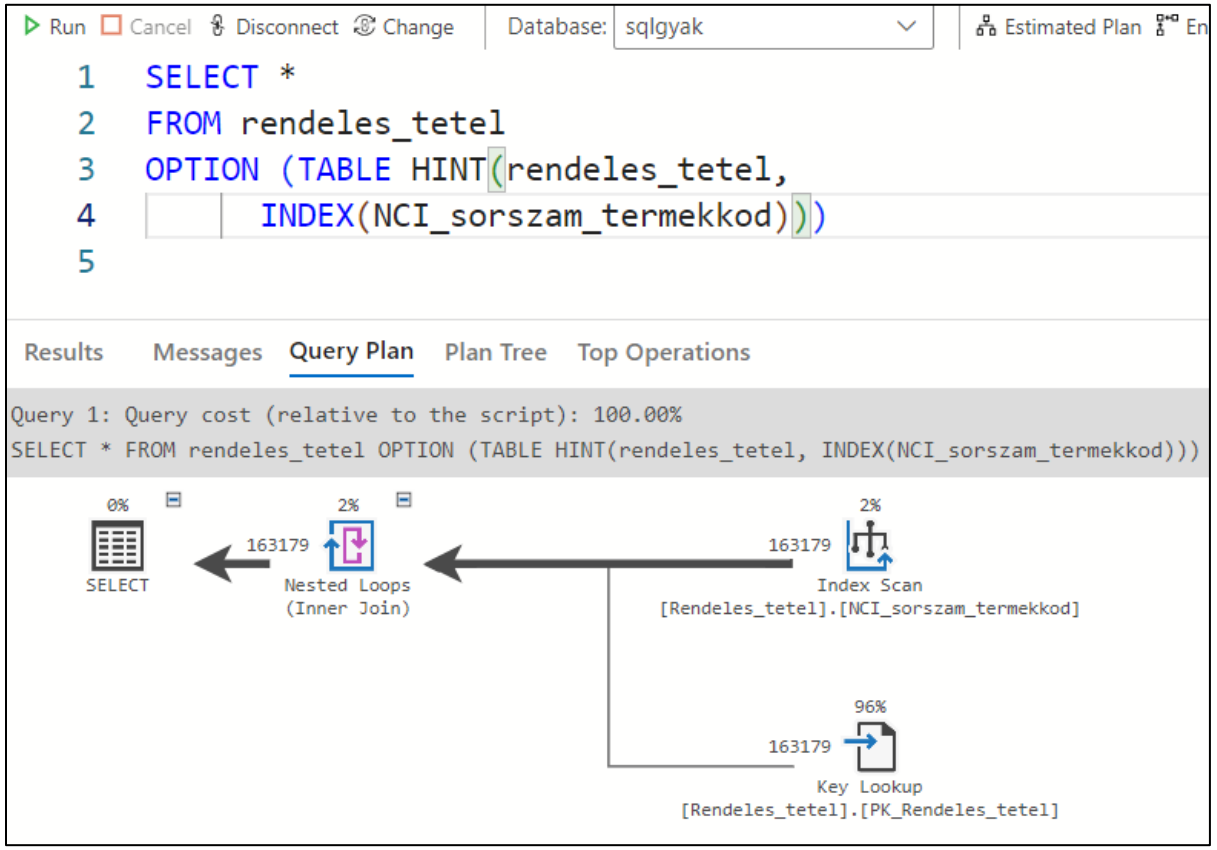
ORDER BY …

**OPTION**
**(TABLE HINT (táblanév,INDEX(indexnév)))**

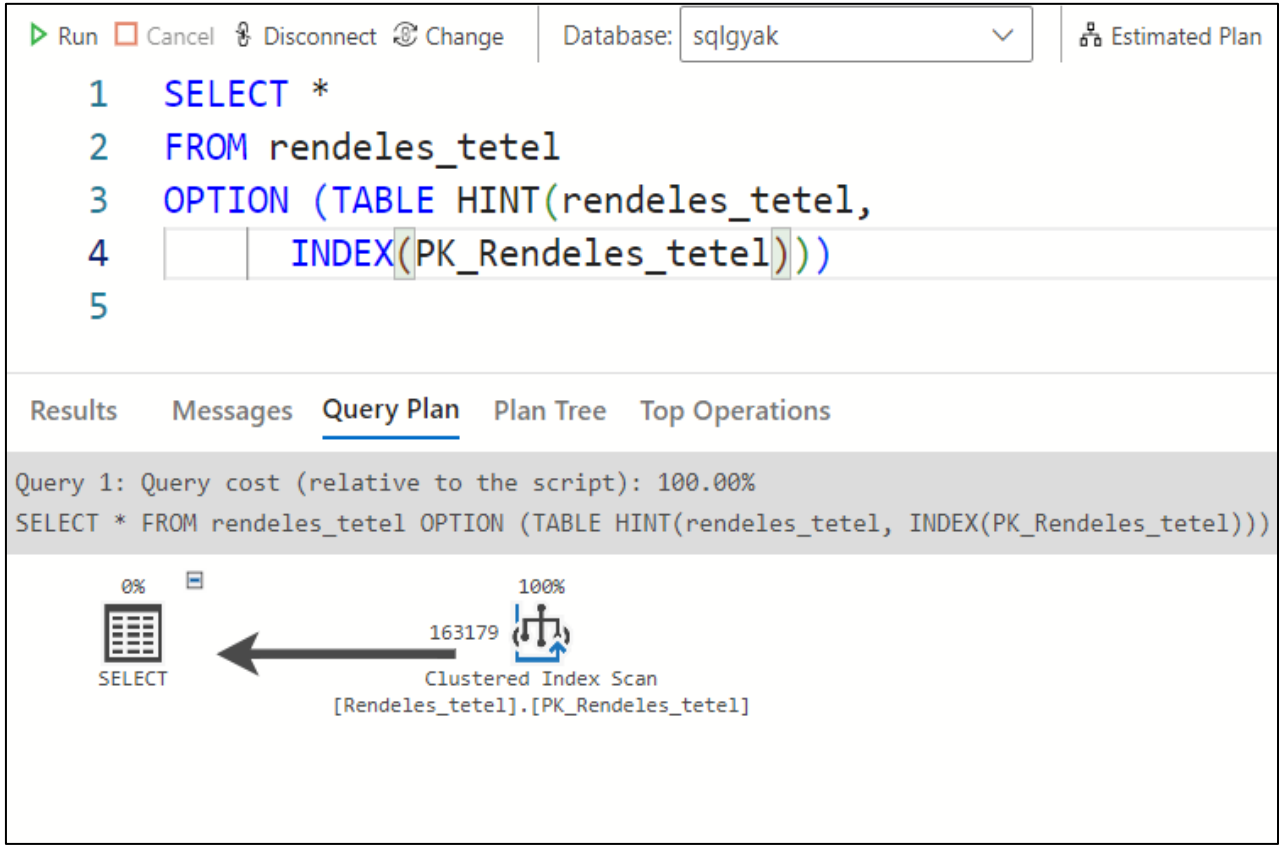
- Többféle HINT létezik:
  - JOIN HINT
  - TABLE HINT
  - QUERY HINT
- Céljuk:
  - Preferált vagy tiltott index
  - JOIN típus, sorrend
  - Táblaelérési mód
  - Párhuzamos végrehajtás

Az SQL Server Query Optimizer által alkalmazott beállításokat csak indokolt esetben bíráljuk felül a HINT-ekkel

# Table HINT példa – index specifikálás




A lekérdezés futtatása egy saját index segítségével




A lekérdezés futtatása Clustered index segítségével

# Indexek – hasznos parancsok

Run | Cancel | Disconnect | Change | Database: sqlgyak | Estimated Plan | Disable Actual Plan | Parse | Enable SQLCMD | To Notebook

```
1  CREATE NONCLUSTERED INDEX [NCI_sorszam_termekkod] ON [dbo].[Rendeles_tetel]
2  (
3      [SORSZAM] ASC
4  )
5  INCLUDE([TERMEKKOD]) -- index létrehozása
6
7  ALTER INDEX NCI_sorszam_termekkod ON rendeles_tetel REBUILD; -- index újraépítése
8
9  ALTER INDEX NCI_sorszam_termekkod ON rendeles_tetel REORGANIZE; -- index újraszervezése
10
11 ALTER INDEX NCI_sorszam_termekkod ON rendeles_tetel DISABLE; -- index letiltása
12
13 ALTER INDEX NCI_sorszam_termekkod ON rendeles_tetel REBUILD/REORGANIZE; -- index engedélyezése
14
15 DBCC SHOW_STATISTICS ('rendeles_tetel', 'PK_rendeles_tetel');  --index statisztikák
16
```

BUDAPESTI CORVINUS EGYETEM

# Az indexek hátrányai

- Tárhelyet foglalnak
- Folyamatos karbantartást igényelnek
  - Minden DML-művelet esetén (automatikus)
  - Újraépítés (Rebuild)
  - Újraszervezés (Reorganize)
  - Tömörítés
  - Index statisztikák
- A DML-műveleteket lelassítják
- Nem ajánlottak
  - Kis táblák esetén
  - Sok NULL értéket tartalmazó oszlopra
  - Olyan oszlopokra, amelynek értékei gyakran változnak

# Adattárolás

# Hogyan tárolódik fizikailag egy adatbázis?

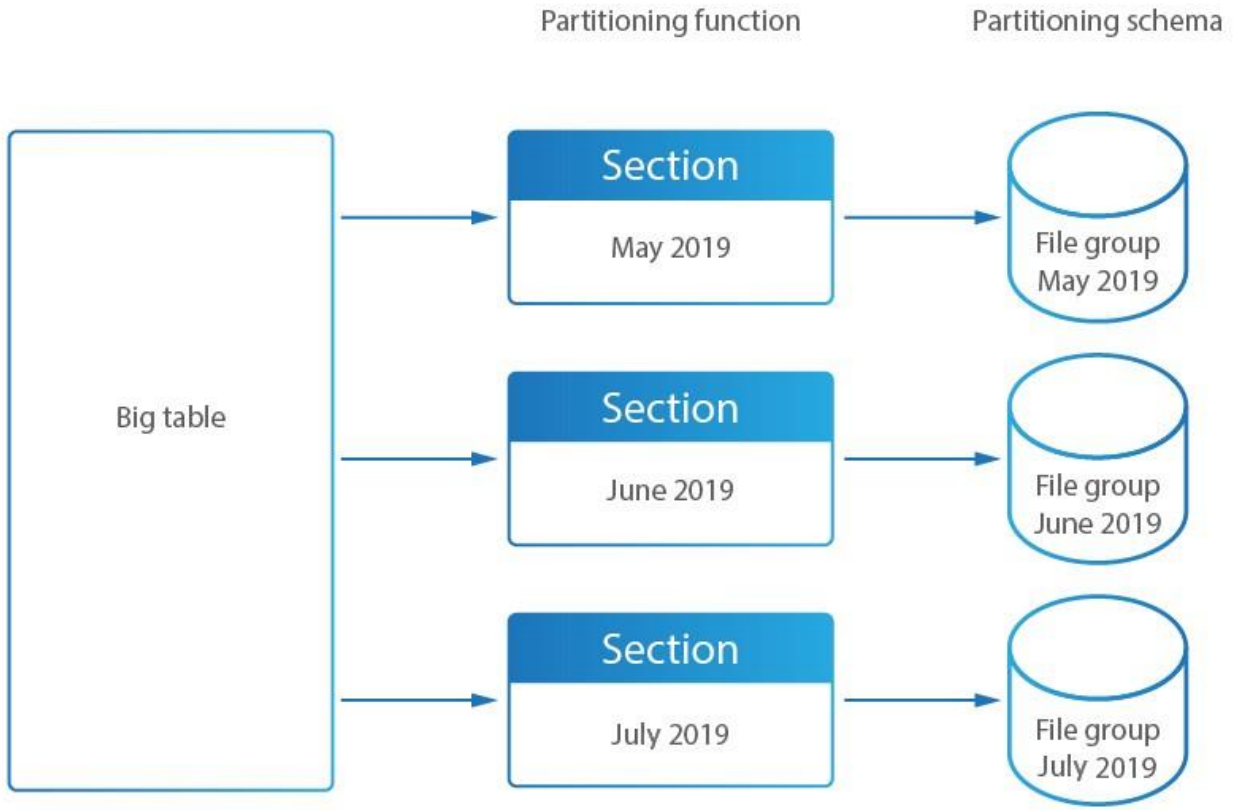
.Az adatbázis tartalma egy vagy több filegroup-ban (fájlcsoportok) tárolódik. Egy fájlcsoport tartalmazhat elsődleges és másodlagos adatfájlt és log fájlt.

- **Elsődleges adatfájl.** Egy **.**MDF kiterjesztésű fájl, ami tartalmazza az egyes adatbázis objektumok (táblák, indexek, nézetek stb.) sémáját és adatait
- **Logfájl.** Egy .LDF kiterjesztésű, automatikusan növekvő fájl, amely a tranzakció adatokat tartalmazza
- **Másodlagos adatfájl.** Egy vagy több .NDF kiterjesztésű fájl, amely az elsődleges adatfájl tárolókapacitásának kiterjesztésére szolgál. (opcionális)

# Partíciók

Nagyméretű táblák vagy indexfájlok önállóan kezelhető részei.

- Az adatokat egy kulcs (fv.) segítségével logikai részekre (partíciók) osztjuk
- A partíciók használata növeli a teljesítményt
- Többféle partíciós stratégia létezik, pl. date range, hash
- Elsősorban adattárházakban és nagyméretű adatbázisoknál használják őket



# Partíciók - példa

```
-- Partíciós fv/stratégia definiálása
CREATE PARTITION FUNCTION DateRangePartitionFunction (DATE)
AS RANGE LEFT FOR VALUES ('2022-01-01', '2023-01-01', '2024-01-01');

-- Partíciós séma definíció
CREATE PARTITION SCHEME DateRangePartitionScheme
AS PARTITION DateRangePartitionFunction
TO ([PRIMARY], [Partition_2022], [Partition_2023], [Partition_2024], [FuturePartitions]);

-- Tábla partícionálása
CREATE TABLE PartitionedTable (
    ID INT,
    Name VARCHAR(50),
    DateColumn DATE
) ON DateRangePartitionScheme(DateColumn);

-- Index készítése a táblához
CREATE CLUSTERED INDEX CX_PartitionedTable ON PartitionedTable(ID);
```

# Tömörítés

.Az SQL Server sor-, illetve lap* szinten tudja tömöríteni a táblákat és az indexeket.

- **Sorszintű tömörítés**. A redundáns információkat csak egyszer tárolja le. Különösen hatékony, ha a tábla sok redundanciát tartalmaz.
- **Lapszintű tömörítés**. Az egész lapot tömöríti egy adott tömörítési algoritmus alapján. Legtöbbször hatékonyabb, mint a sorszintű tömörítés.

*Fix méretű tárolási egység, amely a legkisebb írható/olvasható adatmennyiséget jelenti.

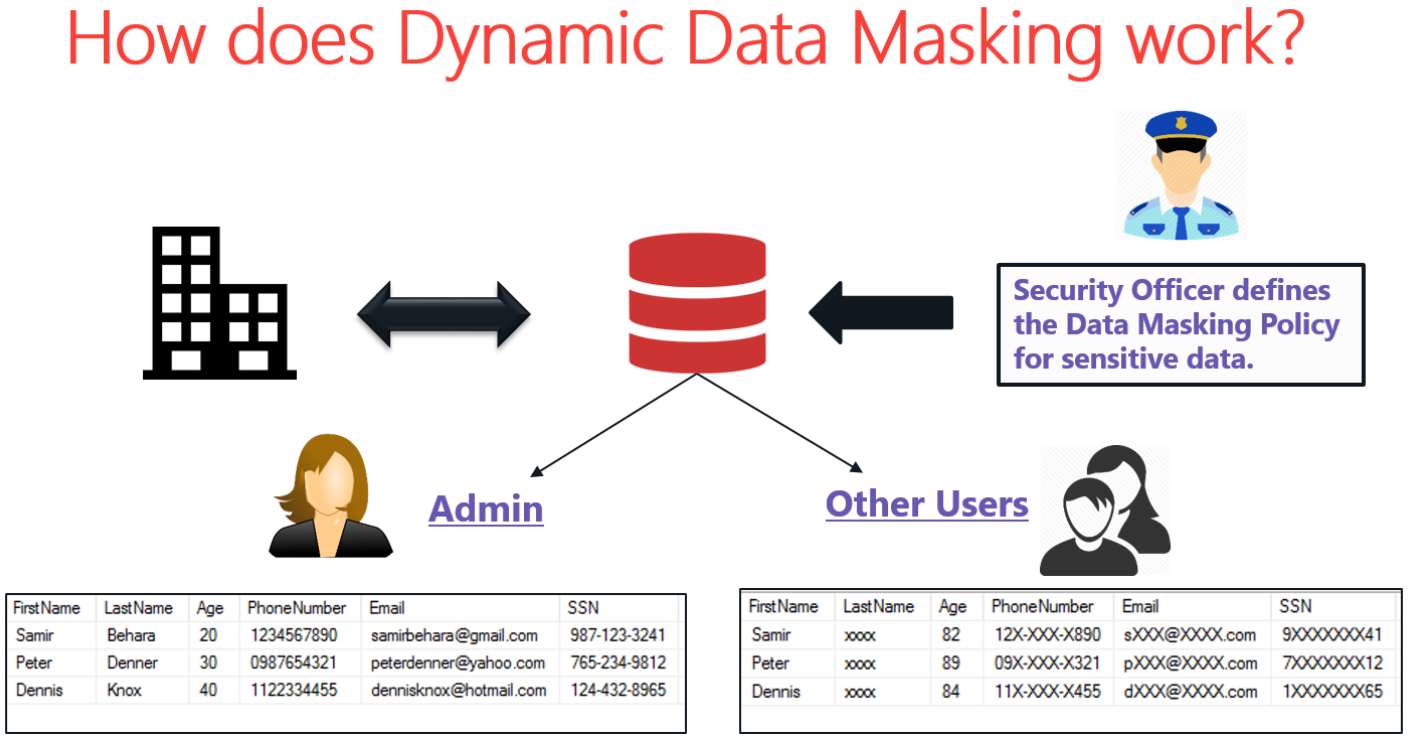
- A lapok rendszerint valamilyen stuktúrába vannak szervezve, pl. halom, B-fa
- A lapoknál nagyobb tárolási egység az extent, amely 8 db egymás melletti lapot jelent
- Hasonló tárolási egység a block, amely fájlrendszertől függően adott menyiségű lapot vagy extent-et jelent

# Titkosítás

## Az SQL Server számos lehetőséget kínál az adatok titkosítására

Fontosabb lehetőségek:

- Dinamikus adatmaszkolás (fv-t használ)
- Oszlop-szintű titkosítás (kulcsokat használ)
- TLS protokoll használata
- Always Encrypted (kulcsot + konfigurációt igényel)
- Transparent Data Encryption (csak Enterprise verziónál)



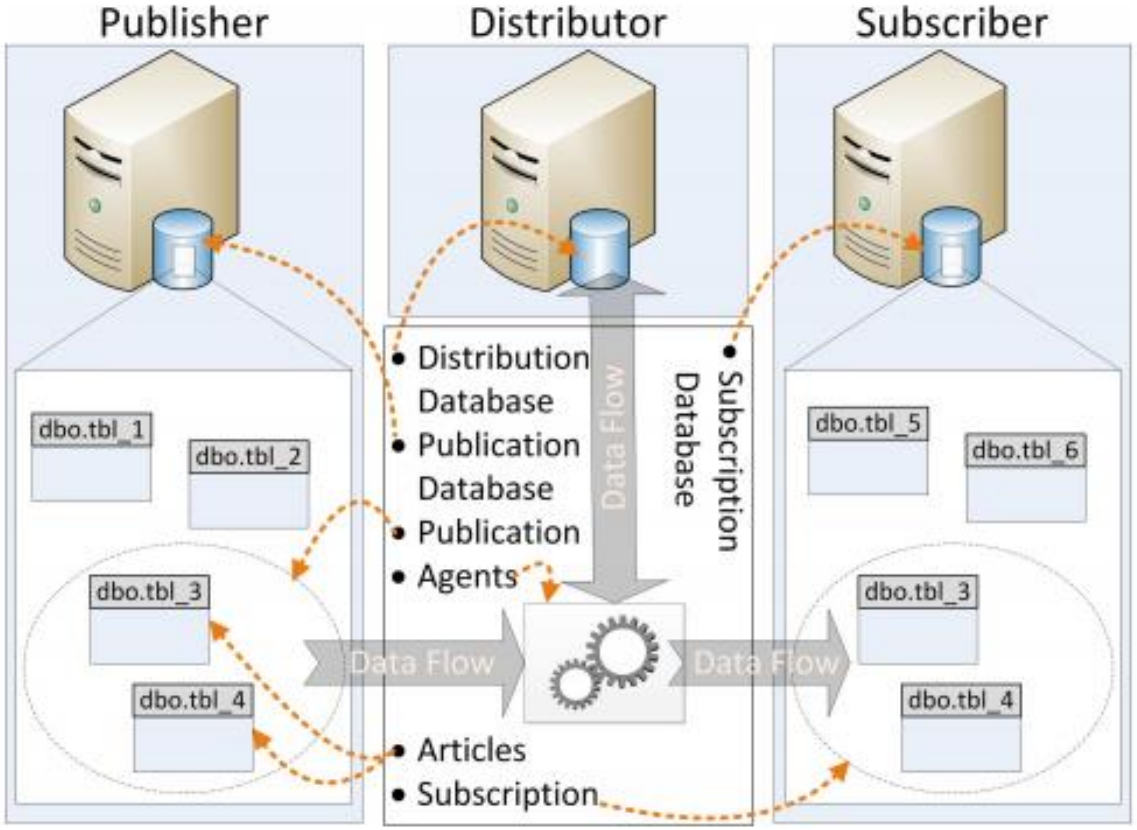

Forrás: Dynamic Data Masking – Altering the masked column – samirbehara

# Replikáció

Egy SQL szerver példányon történő adatok redundáns tárolását, és annak rendszeres szinkronizálását jelenti egy vagy több másik SQL szerver példányon

A replikáció többféle módon megvalósulhat, pl:

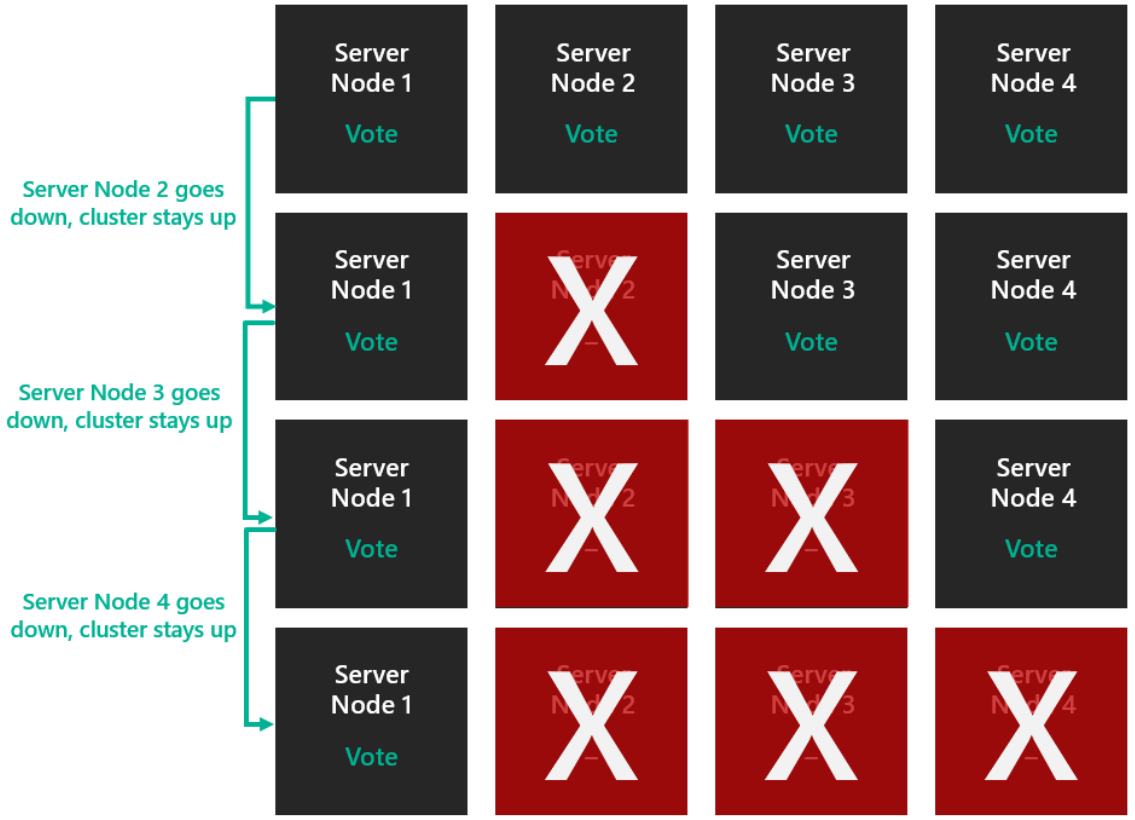
- Snapshot replikáció
- Tranzakciós replikáció
- Peer-to-peer replikáció



# Klaszter

SQL szerverek csoportja, amelyek együttműködése lehetővé teszi a magas rendelkezésre állás és a hibatűrés megvalósítását

- Minden egyes szerveren (node) sql server példány fut
- A szerverek az adatokat egy közös, megosztott tárolón tárolják
- Hiba esetén többségi szavazás (quorum) dönt a folytatásról



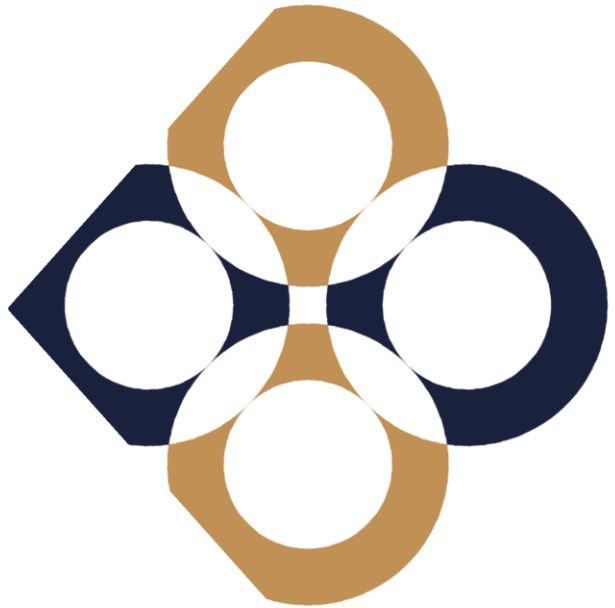

[Understand cluster and pool quorum on Azure Stack HCI and Windows Server clusters - Azure Stack HCI | Microsoft Learn]

# Köszönöm a figyelmet!

BUDAPESTI CORVINUS EGYETEM

# Adatbázisok előadás 06

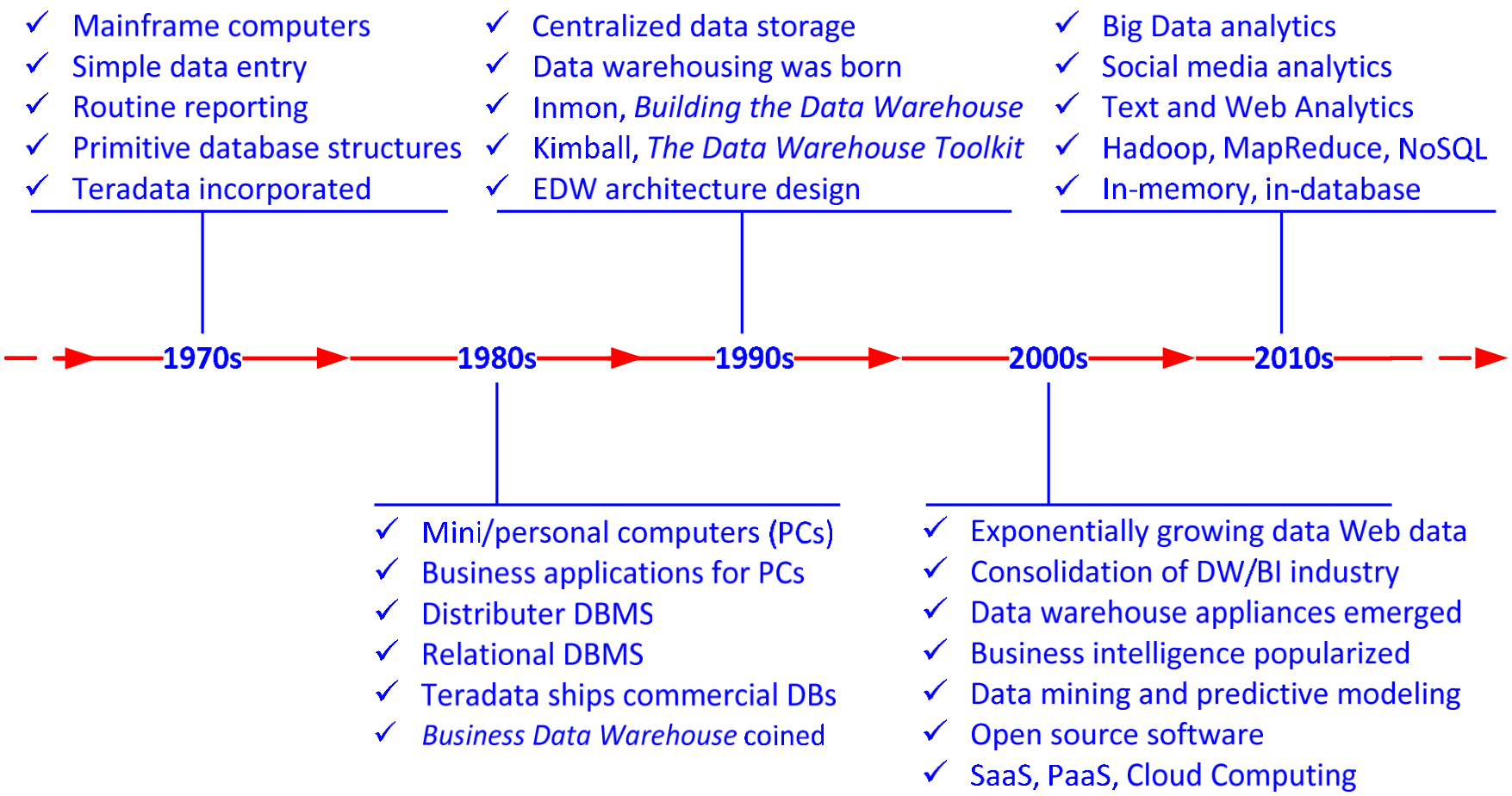
## Adattárházak



# Miről lesz szó?

- *Adattárház és a hozzá kapcsolódó fogalmak*
- *Kimball modellje*
- *Inmon modellje*
- *Kihívások, alternatív adattárház rendszerek*

# Történeti áttekintés - címszavakban



# Vezetői információs rendszerek

- 1980-as évek - Legelső vezetői információs rendszerek
  - Működési célú (tranzakciós, operatív) adatbázisok (OLTP)
  - Előre definiált riportok
  - Egyre több adat --> komoly terhelést jelentett az adatbázis rendszer számára
- Ötlet – készítsünk másolatot az adatbázisról, és azon futtassuk a riportokat
  - A működési és az elemzési célú adatbázisok egyre inkább különváltak
  - Az elemzési célú adatbázisokból fejlődtek ki az adattárházak (DWH, Data Warehouse)

# Miért van szükség adattárházakra?

- ❑ Nehézkes és lassú riportolás
- ❑ Adatminőségi problémák
- ❑ Egységes metaadat kezelés hiánya
- ❑ Elemzési és adatbányászati igények megjelenése
- ❑ Körülményes a riportkészítés több adatforrás esetén
- ❑ Az adatok sokszor különböző formában állnak rendelkezésre

# Milyen elven működnek az adattárházak?

- A különböző forrásokból származó adatokat egy helyre összegyűjtjük, majd
- aggregáljuk olyan mértékben, amilyen léptékben döntéseinket hozzuk, majd
- az üzleti gondolkodásnak megfelelő új struktúrát alakítunk ki, majd
- készítünk olyan eszközt vagy célalkalmazást, amely az elemzéshez szükséges funkciókat biztosítja, végül
- elérhetővé tesszük a vezetők és felhasználók számára

# Adattárház fogalma

Az adattárház a tranzakciós adatok lekérdezési és elemzési célokból speciálisan strukturált másolata (Kimball)

Az adattárház témakör-orientált, integrált, időfüggő, de időben nem változó adatok gyűjteménye, amelyet a cég vezetői döntéshozatalának támogatására használnak (Inmon)

# Adattárház – néhány más elnevezés

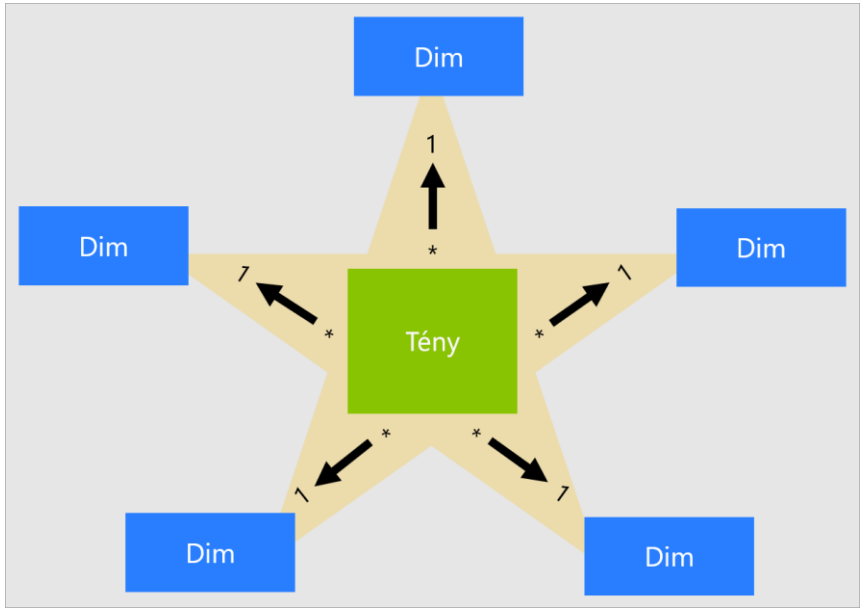
# (OLTP) Adatbázis vs. adattárház

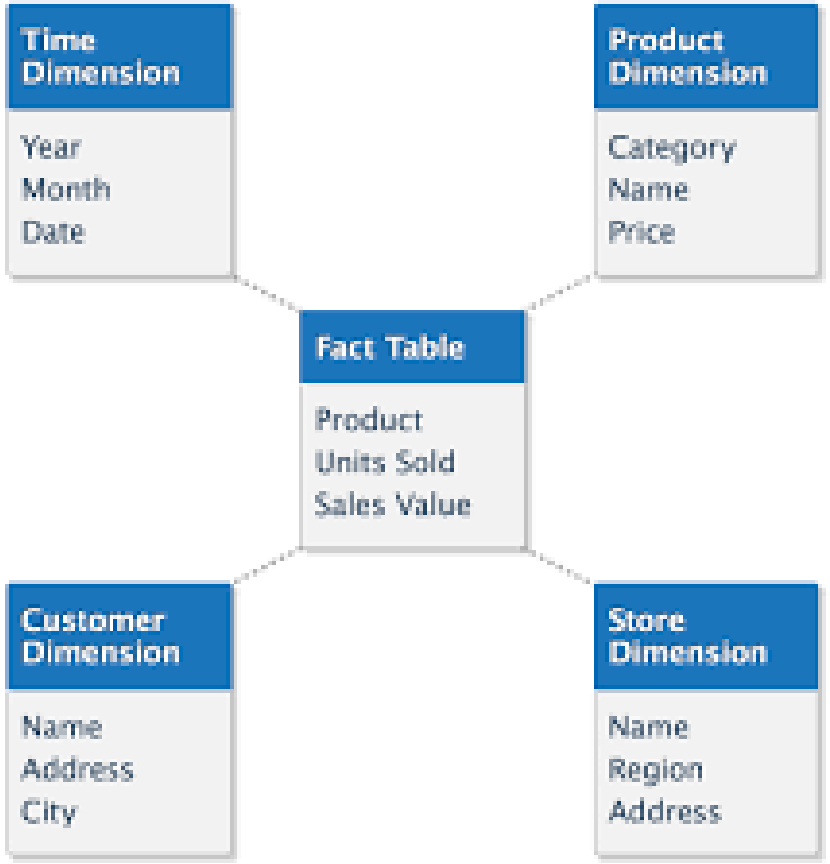
- Sok felhasználó
- Sok kicsi, konkurrens tranzakció
- Rengeteg SQL utasítás (pl. másodpercenként több ezer, esetenként még ennél is több)
- Az utasítások önmagukban leginkább egyszerűek
- A lekérdezések döntő többsége kevés sort érint
- A tranzakciókezelés a fő kihívás, az utasítások végrehajtása általában könnyű
- Elvárt a magasfokú normalizáltság: minél inkább elkerüljük az anomáliákat

- Viszonylag kis számú felhasználó (nevezik őket adatelemzőknek, adatbányászoknak is)
- Viszonylag kis számú, de gyakran igen nehéz lekérdezés
- Nem jellemzők az egyidejű tranzakciók – nem akkora gond a konzisztencia
- Gyakran nem jellemző az adatok online változtatása
- A lekérdezések jellemzően összesített adatokat kérnek
- Jellemzően nem cél az adatok normalizáltsága, hanem az ún. csillag séma (star schema) szerinti az adatmodell

OLTP (Online Transaction Processing): hagyományos (működési célú) adatbázis rendszert jelent

# Csillag (Star) séma



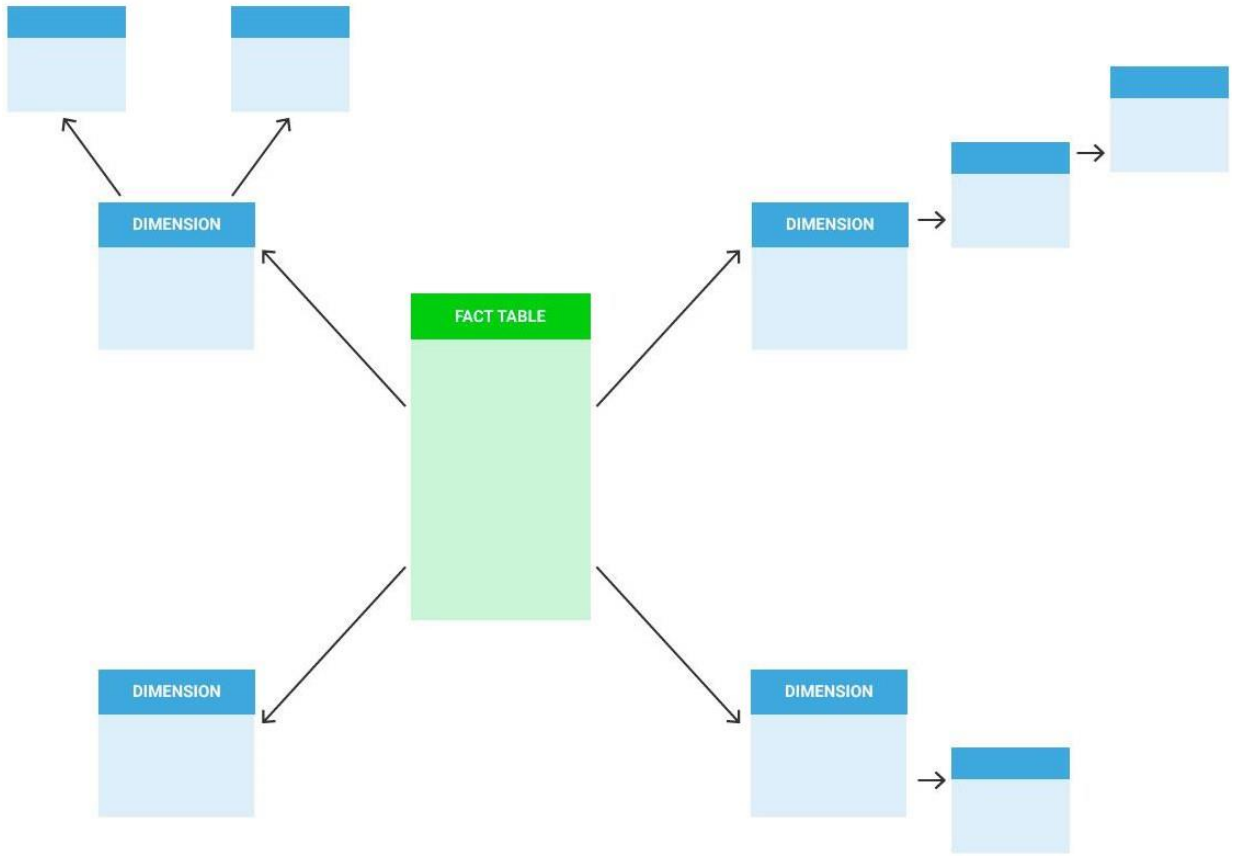



# Tényadatok (ténytáblák) vs. Dimenzionális adatok (dimenziók)

| Szempont | Tényadat | Dimenzió adat |
|---|---|---|
| Cél | Mérőszámok | Leíró adatok |
| Jellemző adattípus | Numerikus | Nem numerikus |
| Előfordulás | Sok rekordban | Kevés rekordban |
| Darabszám | Kevesebb | Több |
| Példa | Eladás összege | Termék kategóriája |

- A ténytáblák tényadatokat és idegen kulcsokat tartalmaznak
- A dimenziótáblák dimenzió adatokat és elsődleges kulcsokat

# Hópehely (Snowflake) séma




A dimenzió táblákhoz újabb dimenzió táblák kapcsolódhatnak

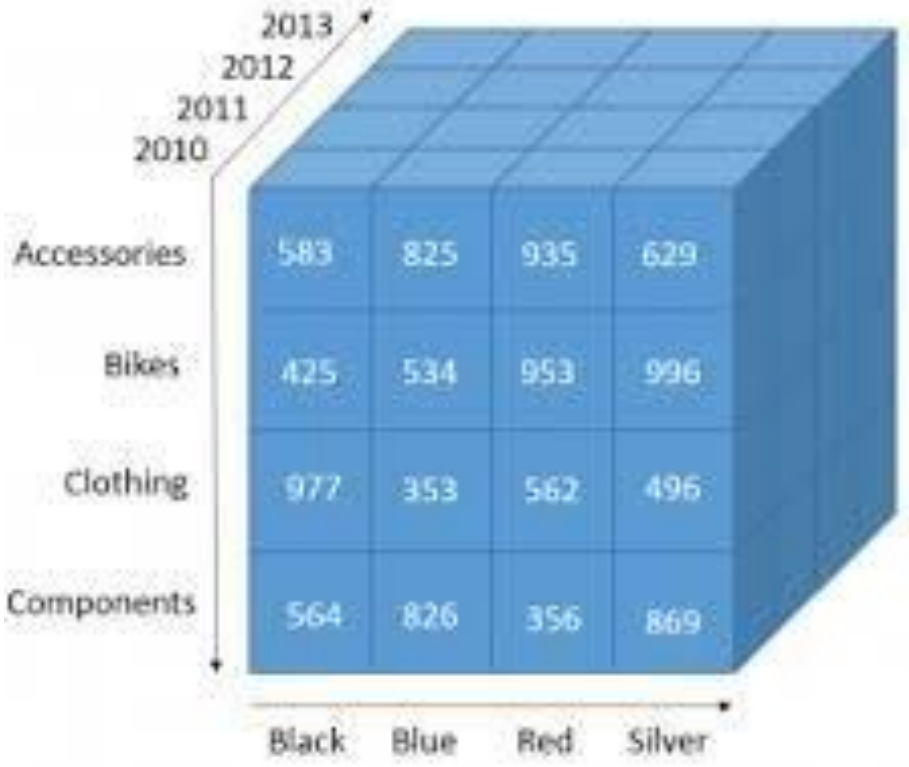
- Kisebb mértékű denormalizáltság
- Kisebb redundancia
- Kisebb helyfoglalás
- Bonyolultabb lekérdezések
- Lassúbb végrehajtás
- Nehezebben áttekinthető

A kép forrása: https://blog.sqldbm.com/bi-data-modeling-with-sqldbm/

# OLAP – Online Analitical Processing

**Codd (1992)**

- Multidimenzionális nézet
- Transzparencia
- Jogosultságok beállíthatósága
- Állandó lekérdezési teljesítmény
- Kliens-szerver architektúra
- Általános dimenzió fogalom
- Egyidejűleg több felhasználó
- Korlátozás nélküli dimenzió műveletek
- Intuitív adatkezelés
- Rugalmas riportozás
- Korlátlan dimenziószám és aggregációs szint



# OLAP vs. OLTP*

| Szempont | OLTP | OLAP |
|---|---|---|
| Cél | Napi működés | Elemzések |
| Felépítés | Relációs adatbázis | Adattárház |
| Lekérdezés | INSERT, DELETE, UPDATE | SELECT |
| Tábla | Normalizált | Nem normalizált |
| Forrás | Egy forrásrendszer | Több OLTP forrásrendszer |
| Válaszidő | Rövid | Nagy |

* Nem véletlen, hogy az adatbázis vs. adattárház összehasonlításhoz hasonló szempontok vannak itt is. Az adatbázisokat sokszor az OLTP, adattárházakat pedig az OLAP névvel is azonosítják

# Adatpiac (Data Mart) – nincs egységes meghatározás

Az adatpiac nem normalizált, aggregált adatokat tartalmaz, és valamely szervezeti egység működési, felhasználói követelményei által meghatározott (Inmon, 2002)

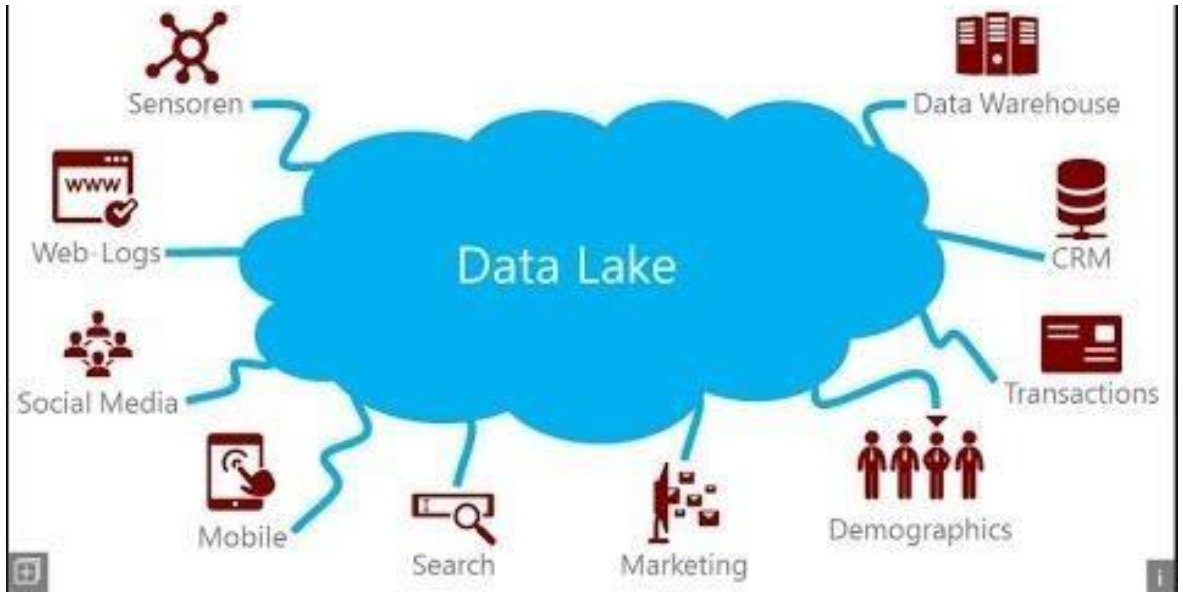
Az adatpiac az adattárház egy tárgyterülethez (pl. marketing) köthető része (Turban, 2014)

Az adatpiacok az adattárházakhoz hasonló adatkezelési képességekkel rendelkeznek, de egy-egy szervezeti egység speciális információs igényeinek megfelelően optimalizáltak (Wikipédia)

Az adatpiac lehet egy önállóan létező, az adattárháztól független megoldás, vagy annak része

# Adattavak (Data Lake)

Nagyvállalati szintű adatmenedzsment-platform, amelyen a különböző forrásokból származó adatok natív formátumukban érhetők el elemzésre



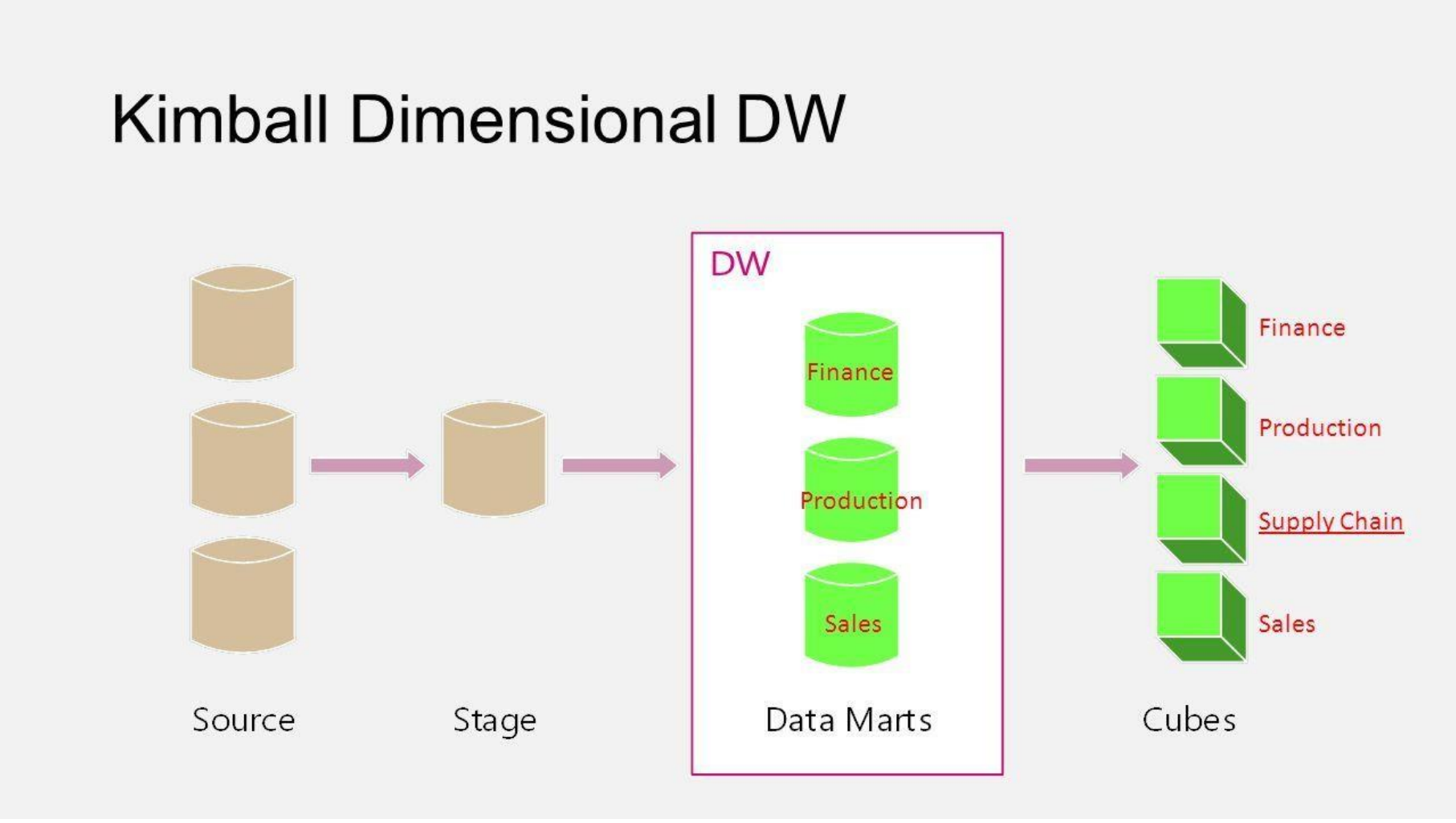

- Big Data rendszerekre jellemző
- Központosított adatkezelés
- Az adattárházaknál rugalmasabb struktúra
- Strukturált és nem strukturált adatokat is tartalmazhatnak

A kép forrása: https://www.youtube.com/watch?v=a3nezIjxI-E

# Adattavak vs. adattárházak

| Data lake | Data warehouse |
|---|---|
| Az adatok tárolásának célja előre nem definiált | Előre definiált tárolási cél |
| Az adatok nyers formában tárolódnak | Az adatok lekérdezésre alkalmas formában tárolódnak |
| Adattudósok, adatelemzők használják | Üzleti felhasználók használják |
| Feltörekvő technológia | Kidolgozott technológia |
| NoSQL lekérdezések | SQL lekérdezések |
| Gyors válaszidő | Lassú válaszidő |
| Alacsony költségű tárolás | Magas költségű tárolás |

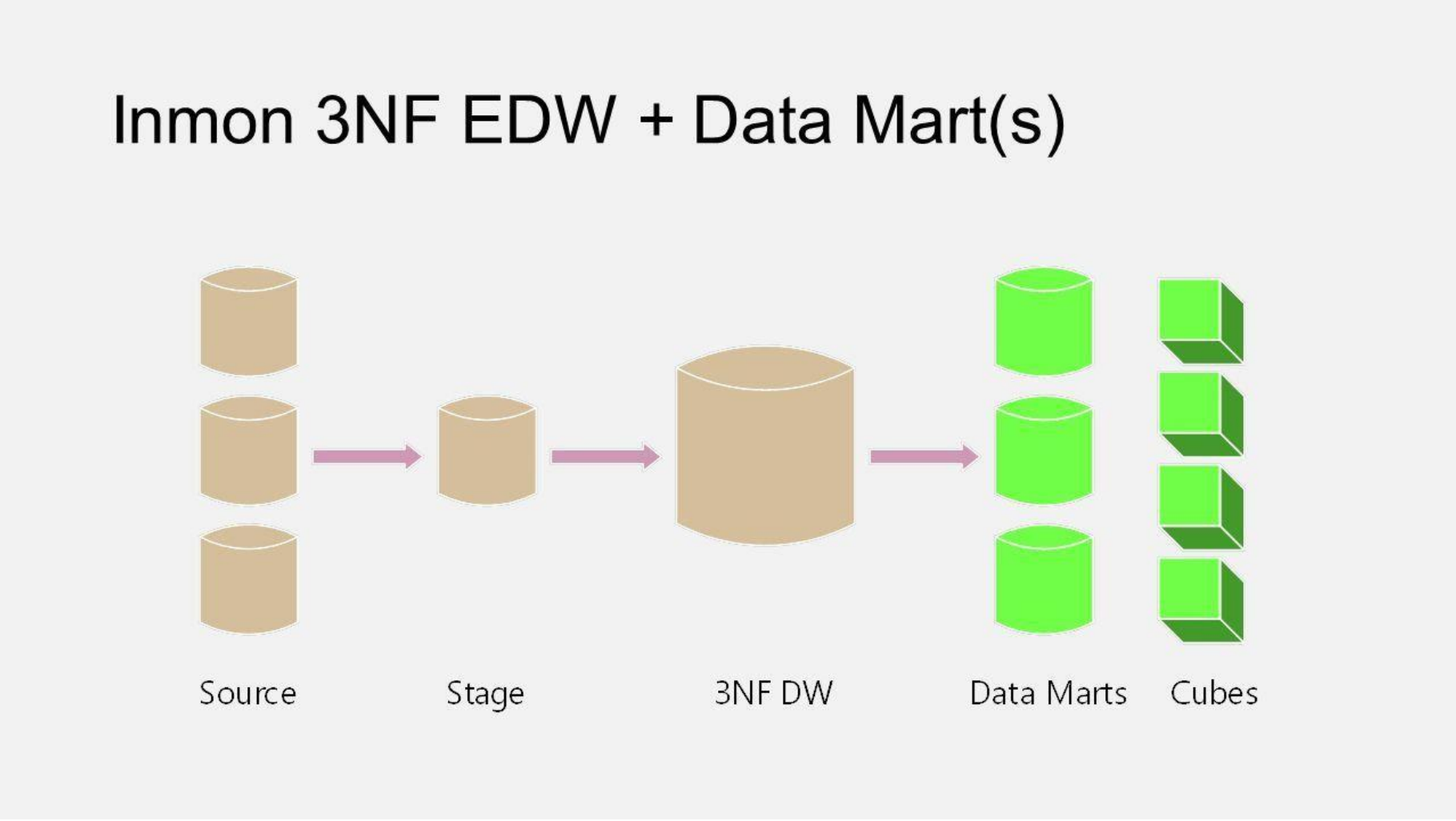
# Kimball adattárház modellje – alulról fel megközelítés



# Kimball adattárház modellje – alulról felfelé megközelítés

- A fókusz az adatpiacokon van
- Az adatpiacok tartalmaznak elemi és összegzett adatokat is
- Az adatpiacok csillag szerkezetűek
- Az architektúra legfontosabb részei a stage terület és az adatpiacok

# Inmon adattárház modellje – fentről le megközelítés



# Inmon adattárház modellje – fentről le megközelítés

- A fókusz az adattárházon (DW) van
- Az adattárház elemi adatokat tartalmaz normalizált formában
- Az adatpiacok összegzett adatokat tárolnak téma specifikus, dimenzionális modellben
- Az architektúra fontosabb rétegei a staging area, a DW, és az adatpiacok
- A felhasználó lekérdezhetnek akár az adatpiacokból, akár az adattárházból is

# Kimball vs. Inmon

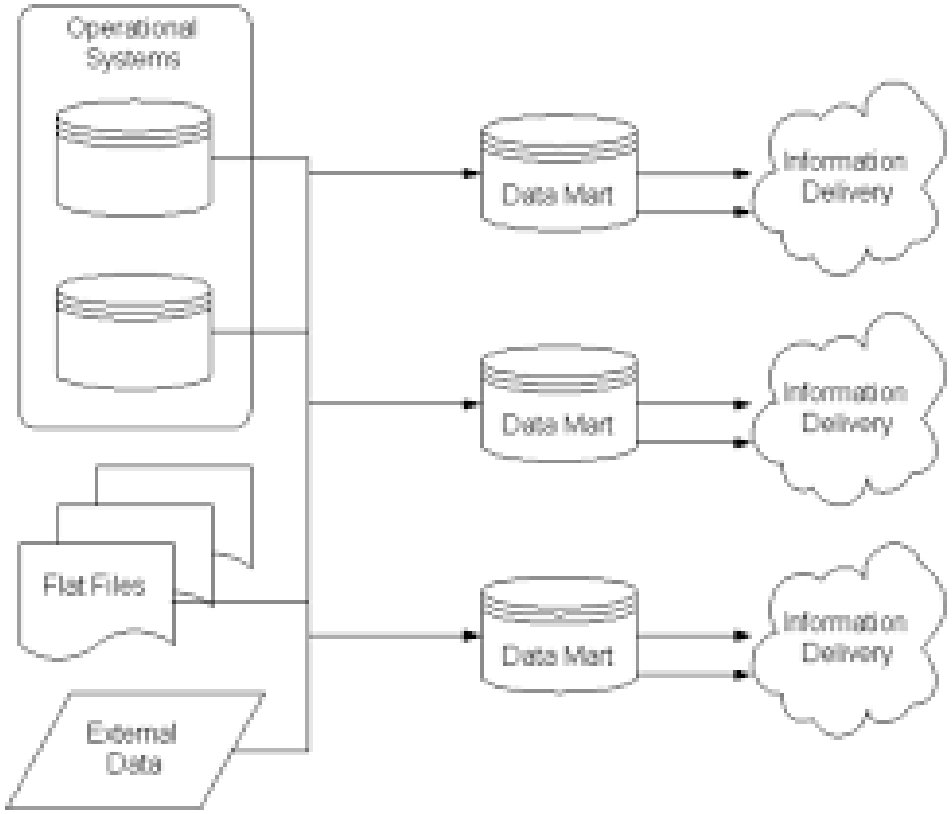
Kimball modellje a megfelelő, ha

- a felhasználók IT területről kerülnek ki
- inkább taktikai döntések szükségesek
- a forrásrendszerek viszonylag stabilak
- minél előbbi eredményt szeretnénk elérni, kis kezdeti befektetéssel és csapattal
- az adatok különálló üzleti területekről jönnek
- a változások köre limitált

Inmon modellje a megfelelő, ha

- a felhasználók nem IT szakemberek
- stratégiai döntések vannak túlsúlyban
- a forrásrendszerek gyakran változnak
- több idő, pénz és nagyobb létszámú csapat áll rendelkezésre
- vállalati szintű adatintegráció szükséges
- a változások köre bővülhet.

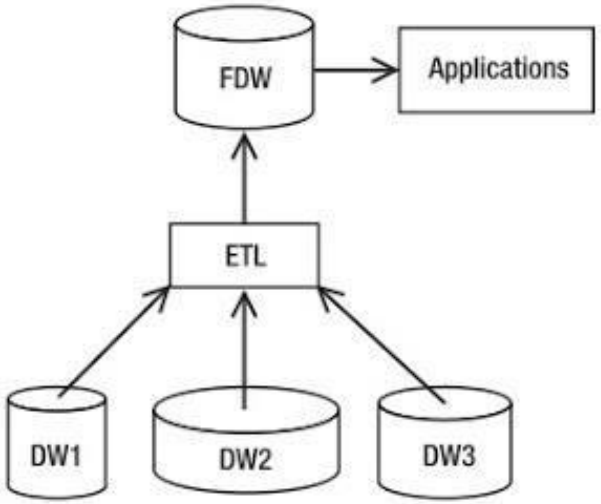
# Alternatív adattárház architektúrák
## Példa 1: független adatpiacok (independent data marts)




- Nincs központi adattárház
- Nincs központi jogosultság rendszer
- Az adatok közvetlenül a forrásrendszerekből érkeznek
- Biztonságosabb
- Kisebb méretű rendszereknél alkalmazható

A kép forrása: https://www.researchgate.net/figure/Data-flow-when-using-independent-data-marts_fig6_34267502

# Alternatív adattárház architektúrák
# Példa 2: egyesített adattárház (federated data warehouse)




*A federated data warehouse from several data warehouses*

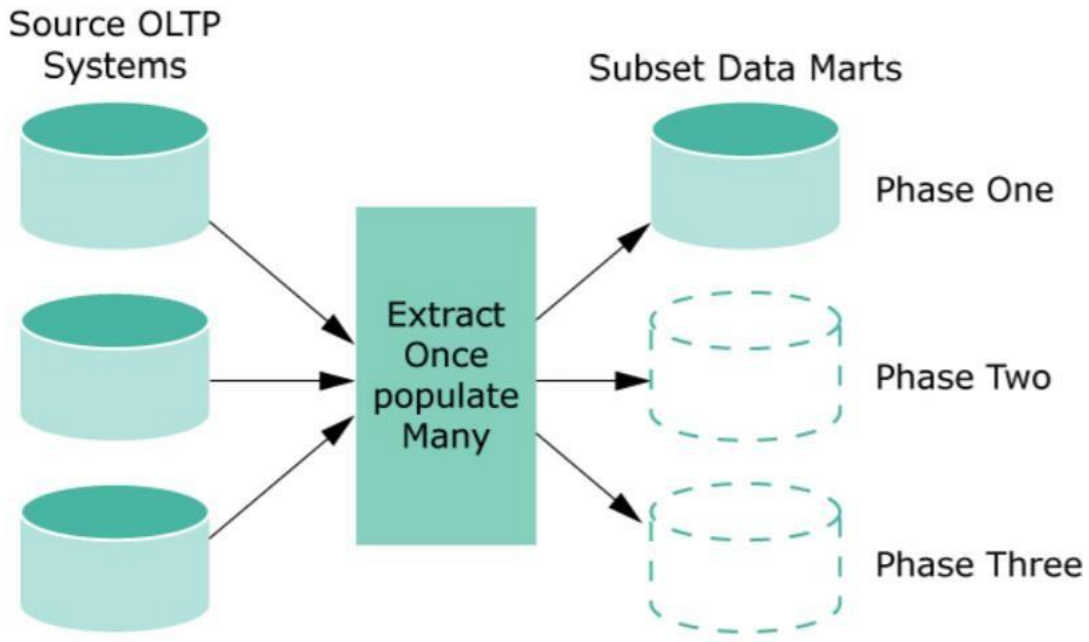
Több adattárházat egyesít

- Pl: nagyobb cég, telephelyek több országban, mindegyiknek saját adattárház
- Közös üzleti szabályok, egyszerű lekérdezés
- Nehéz technikai megvalósítás

A kép forrása: http://blog.umy.ac.id/yusufha/2016/09/21/2-data-warehouse-architecture/

# Alternatív adattárház architektúrák

## Példa 3: Inkrementálisan felépített adatpiacok (incremental architected data marts)




- Az adatpiacok több fázisban jönnek létre
- Egyszeri adatkinyerés, többszöri felhasználás

A kép forrása:
http://www.just.edu.jo/~mzali/courses/Fall14/Cis330/handouts/Successful%20Data%20Warehouses.pdf

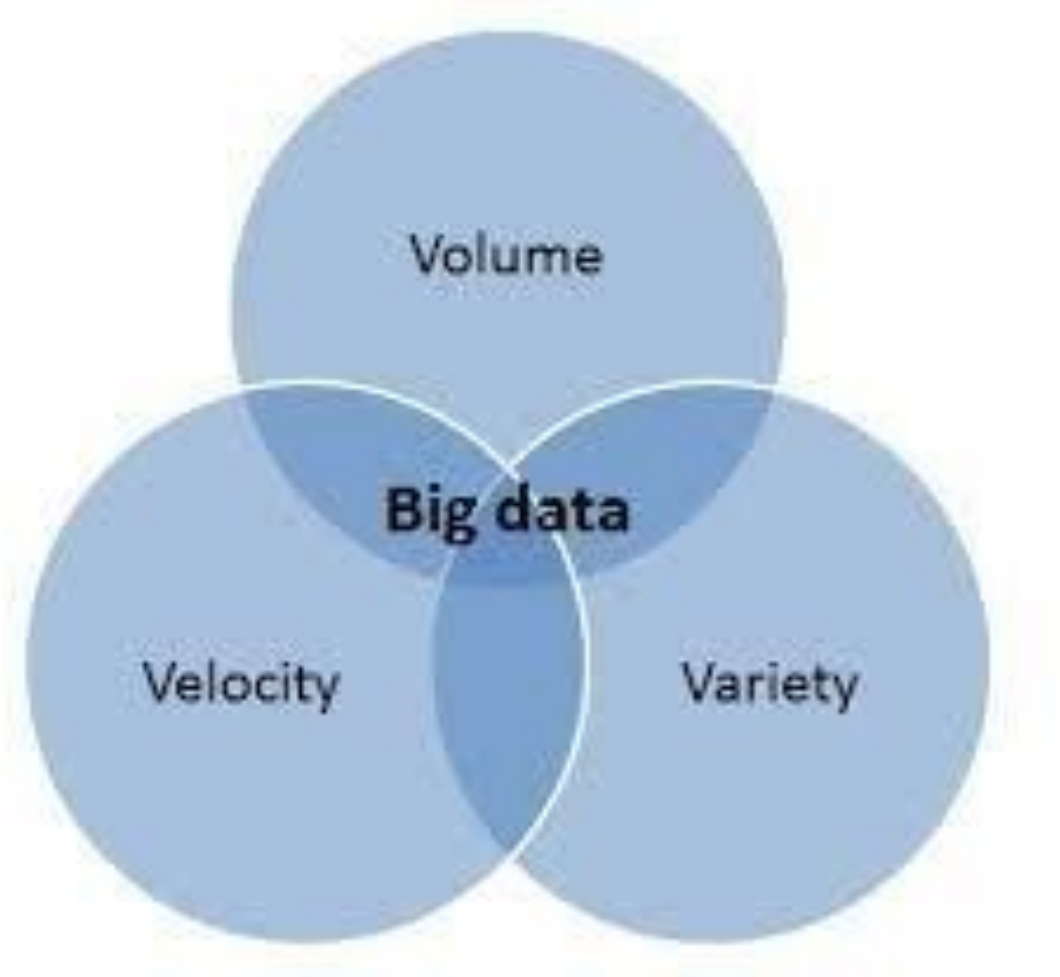
# Adattárházak – új kihívások

- Növekvő adatmennyiség
- Adatfrissítés szinte azonnal
- Gyakori változások az adatmodellben
- Lekérdezések futtatása a memóriában

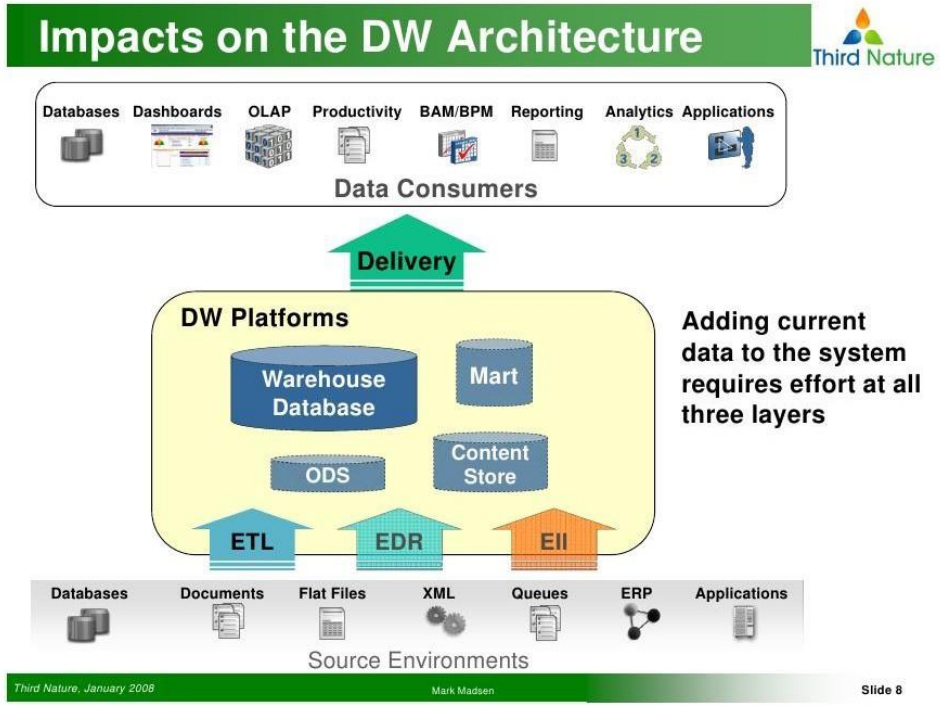
# Egyre növekvő adatmennyiség - Big data

Két fő probléma

- ETL-folyamatok
- Számítások



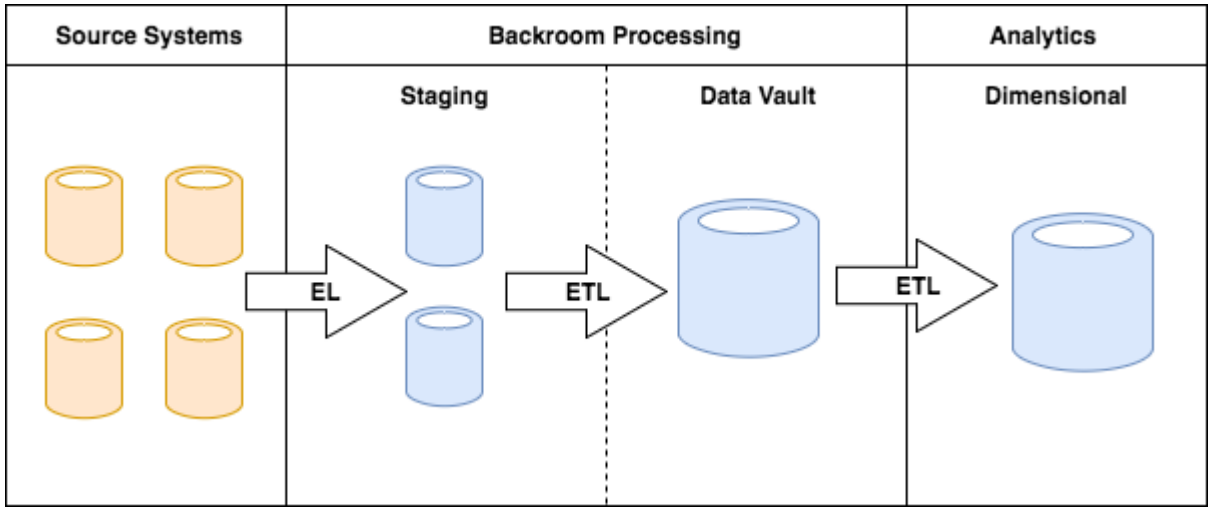
# Gyakori adatfrissítés - Valós idejű adattárház




- Közel valós idejű adatok
- Gyors válaszidő
- Nagy számú felhasználói kérés
- Flexibilis, ad-hoc riportolási lehetőség

A kép forrása: https://www.slideshare.net/mrm0/how-real-time-data-changes-the-data-warehouse

# Gyakori adatmodell változás - Data vault



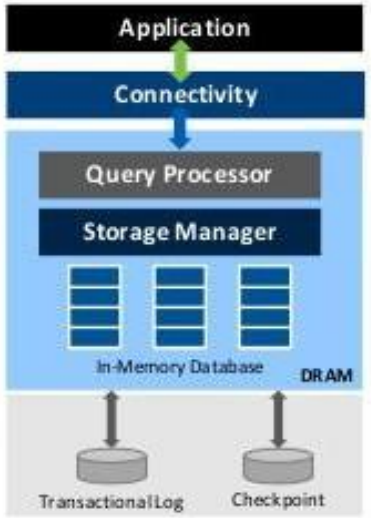

Az adatmodellt metamodell szintjén kezeli

- Speciális fogalmak
  - Üzleti kulcs (hub)
  - Üzleti kulcs tranzakció (link)
  - Üzleti kulcs történet (sat)

# In-memory adatbázisok

## + In-Memory Database Technology

- Extremely Fast Transaction processing
  - Entire database resides in computer's memory
  - Powered by special algorithms and data structures that are highly optimized for in-memory computing
  - Hundreds of thousands of transactions per second
- Short and Predictable Response times
  - Optimized for fastest transactional processing with the shortest response times measured in microseconds
  - The improved response times fuel High Throughput.




A kép forrása: https://www.slideshare.net/Altibase/solve-big-data-problem-the-in-memorydatabase-solution-17179969

BUDAPESTI CORVINUS EGYETEM

# Az előadás diák egy része, illetve azok alap ötlete dr. Kő Andrea: „Adattárházak áttekintés” előadásából származik

# Köszönöm a figyelmet!