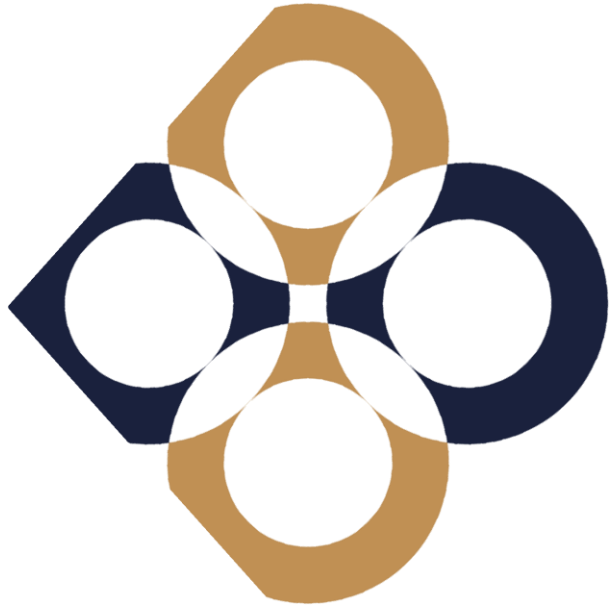
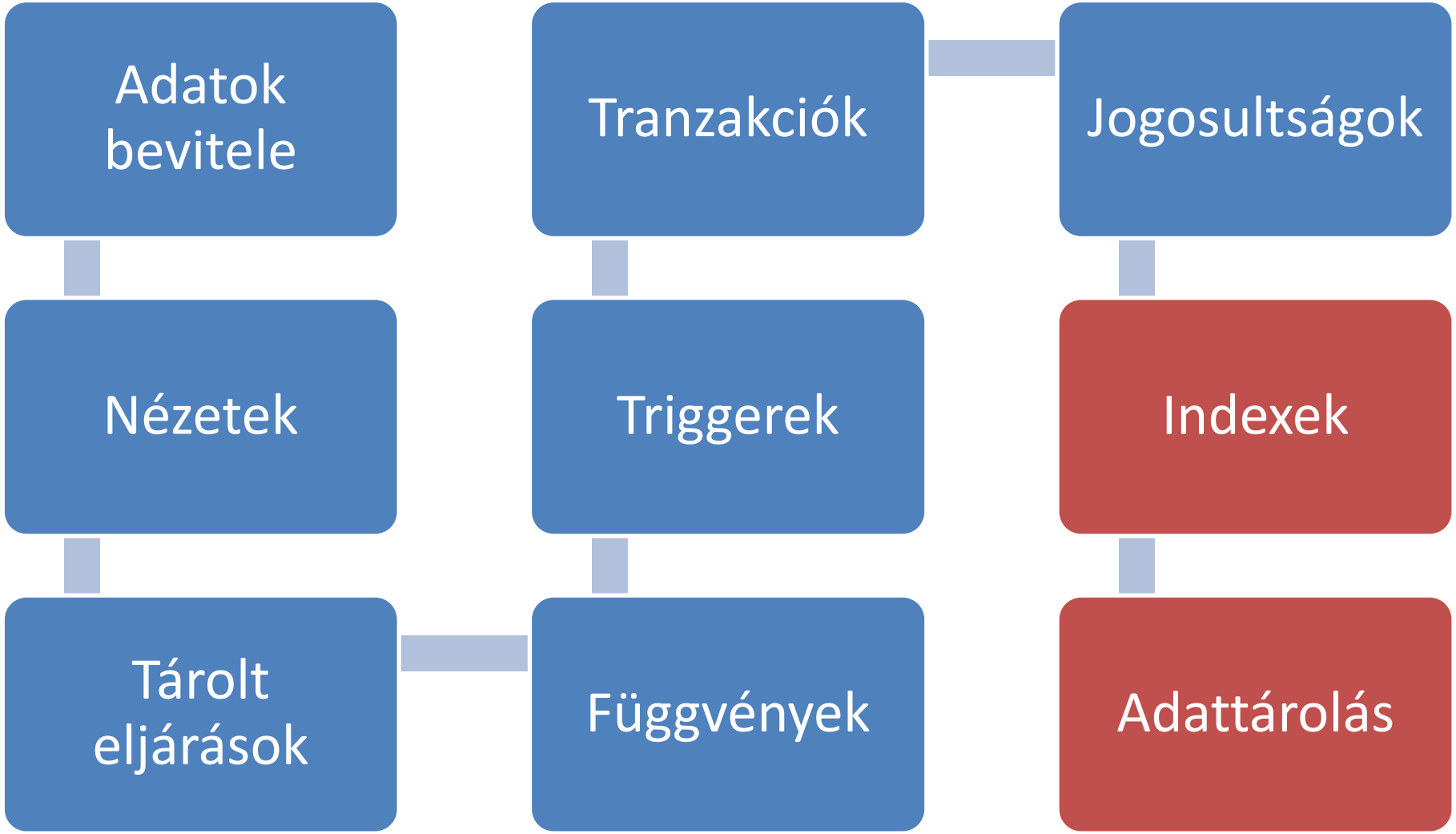
BUDAPESTI CORVINUS EGYETEM

Adatbázisok előadás 05



Fejlesztési és konfigurálási feladatok

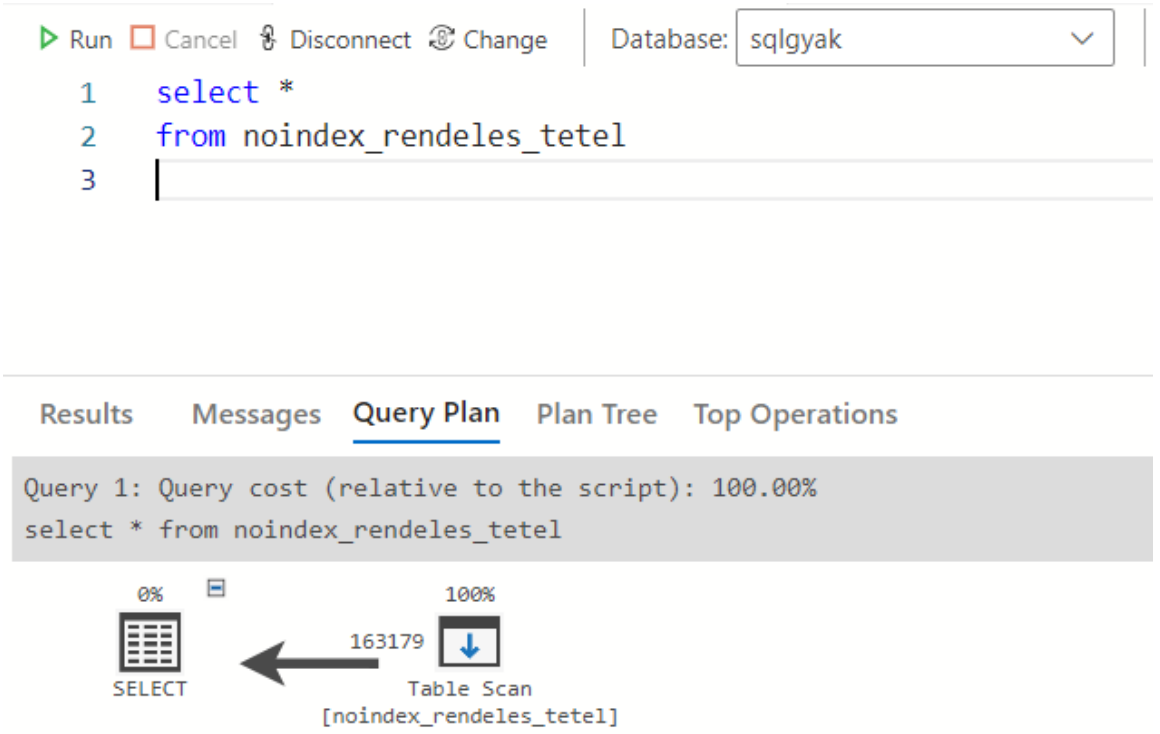


BUDAPESTI CORVINUS EGYETEM

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.

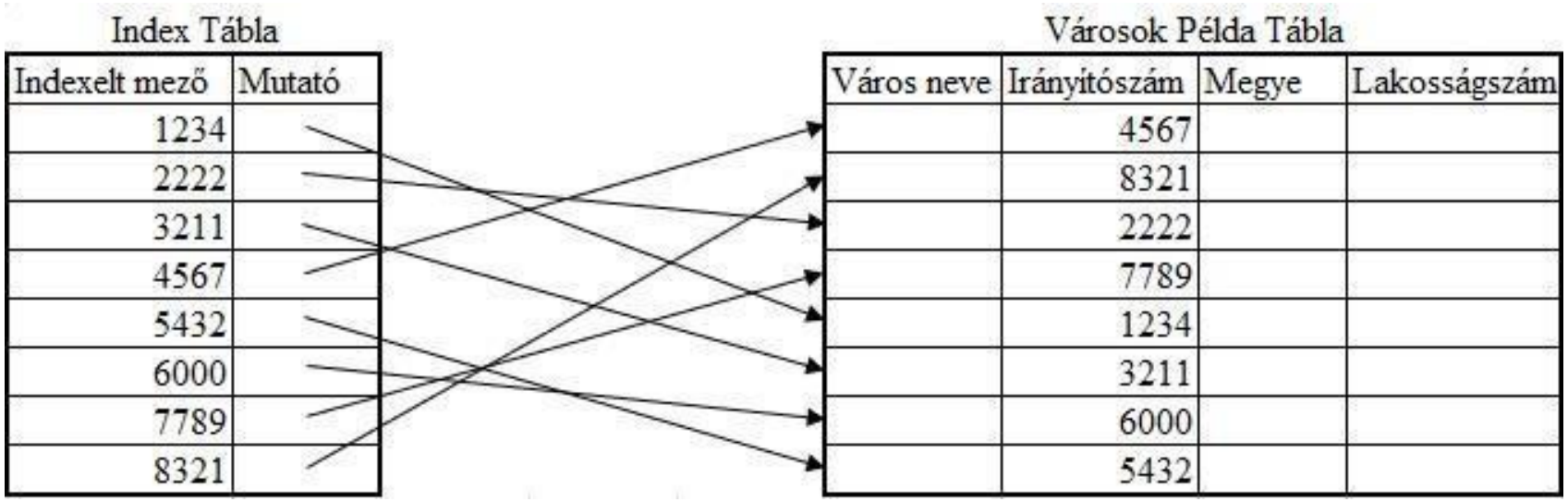
Az indexmutató egy blokkra mutat.
A blokkon belül a keresés szekvenciális

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.



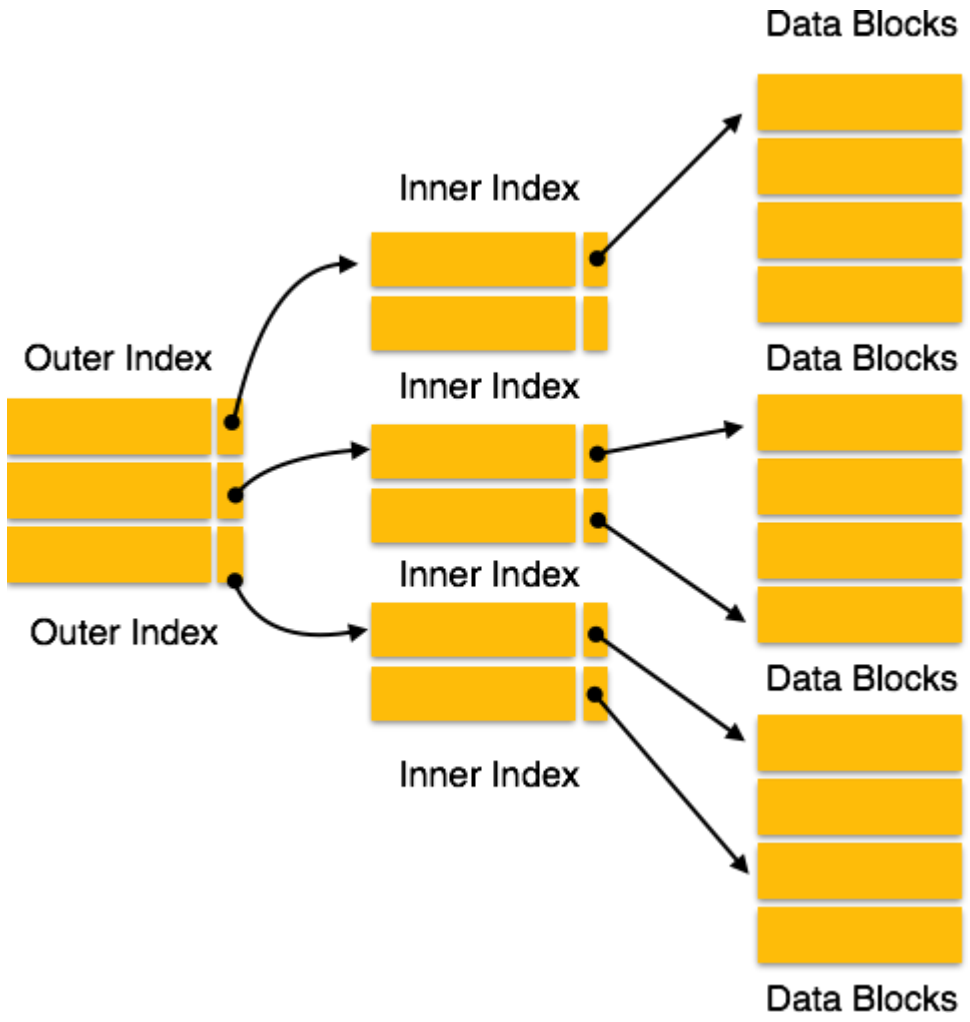
Index Tábla

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		

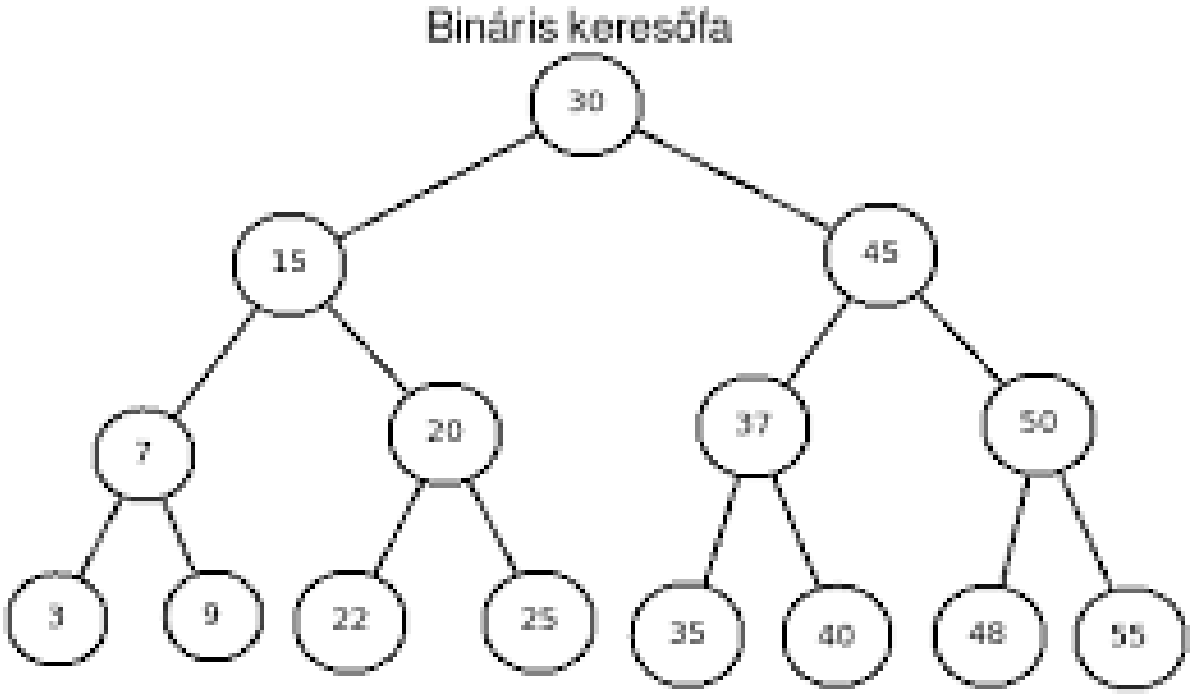
Többszintes indexek



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

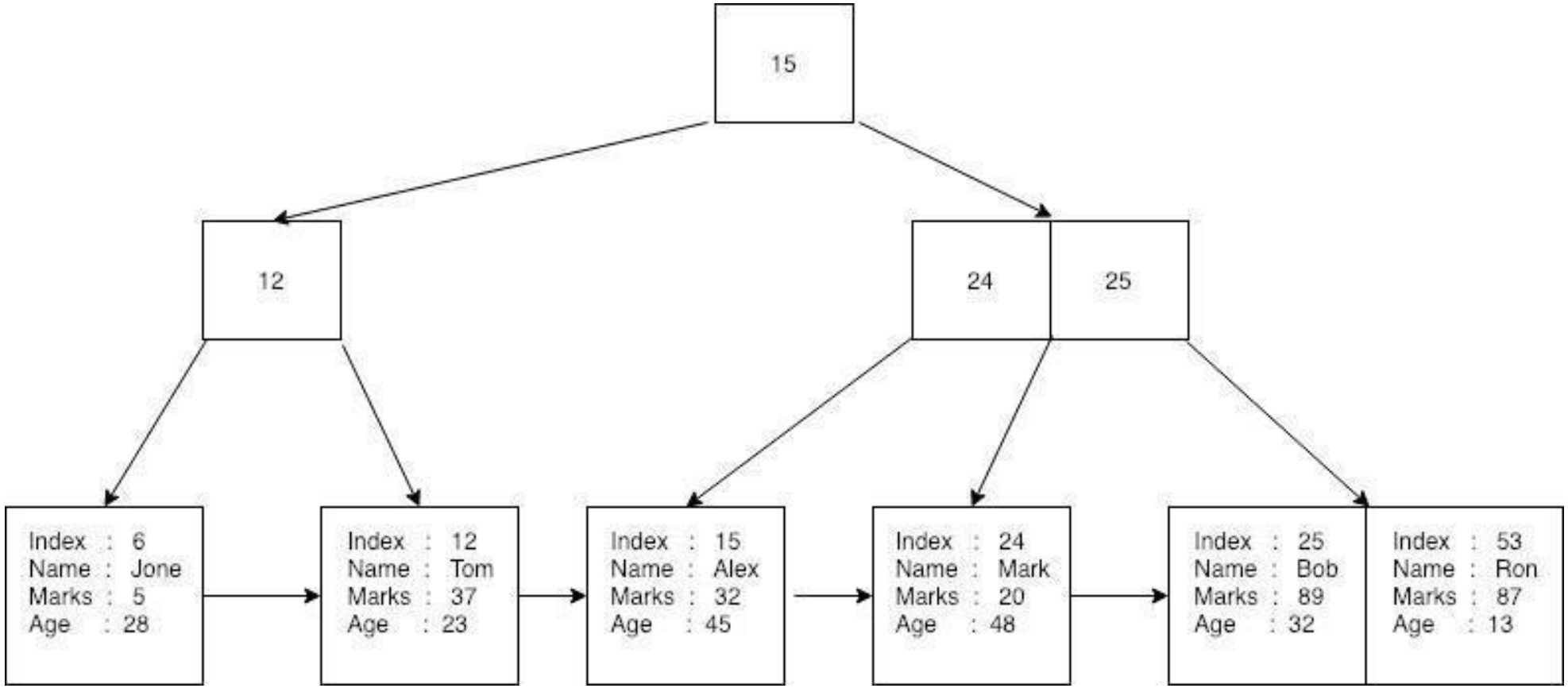
BUDAPESTI CORVINUS EGYETEM

B-fák tulajdonságai

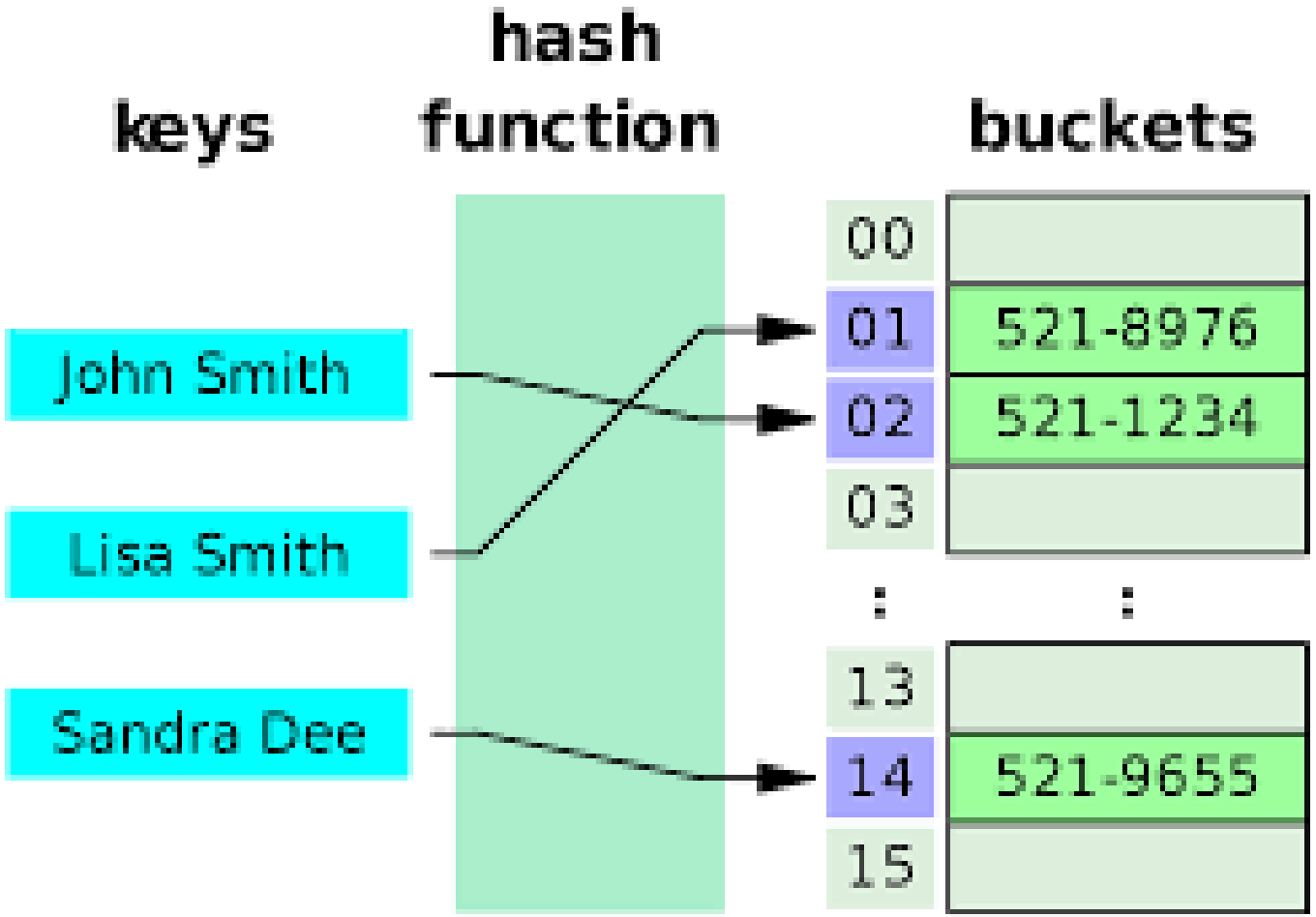
- A gyökértől a levelekig vezető utak hossza egyforma
- Az indexek a B-fa csomópontjaiban helyezkednek el
- Az adatok helyét jelző mutató csak a levelekben található
- A struktúra lehetővé teszi a soros és a random elérést is

BUDAPESTI CORVINUS EGYETEM

B-fák (kiegyensúlyozott keresőfák)

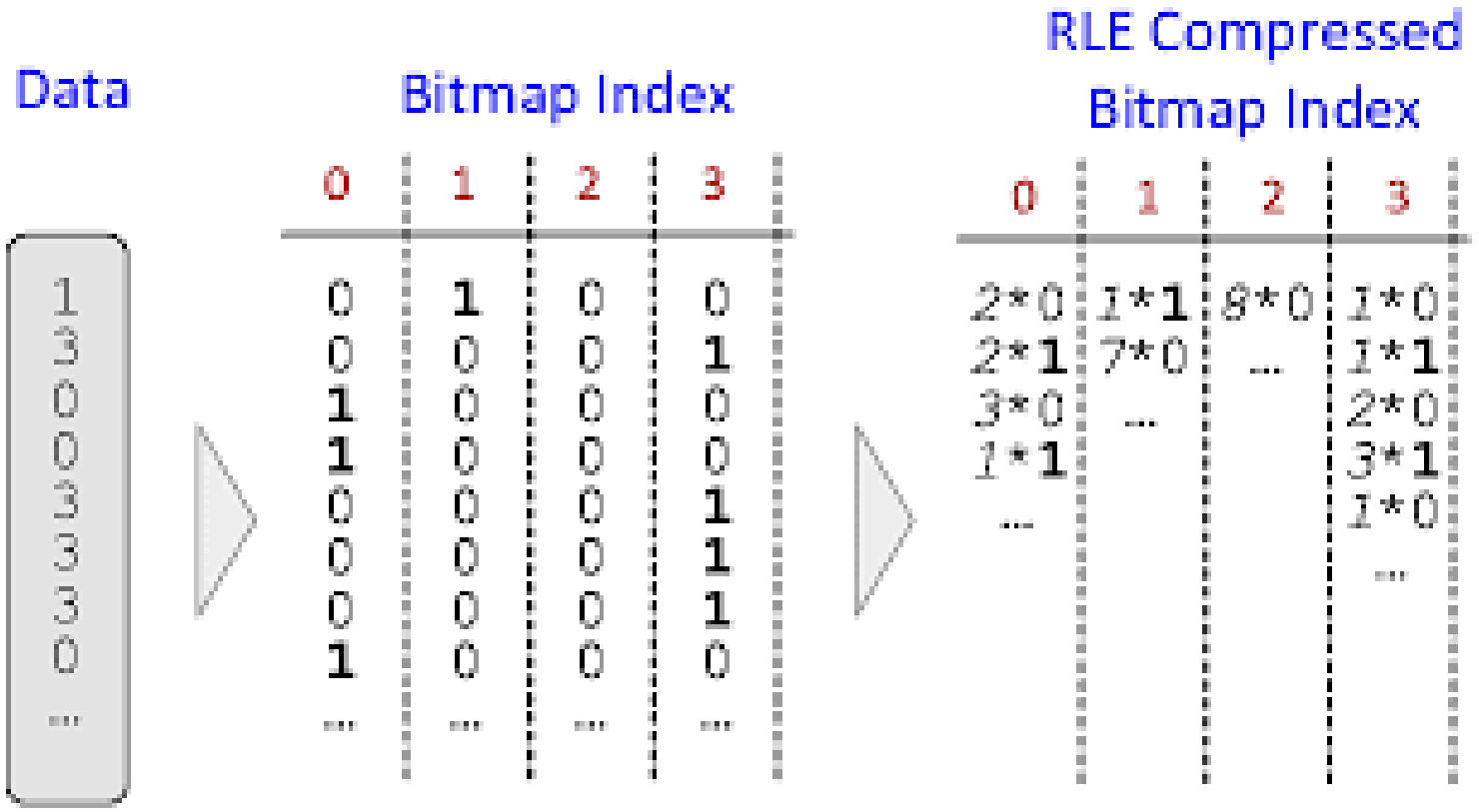


Hash-alapú indexek



- 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

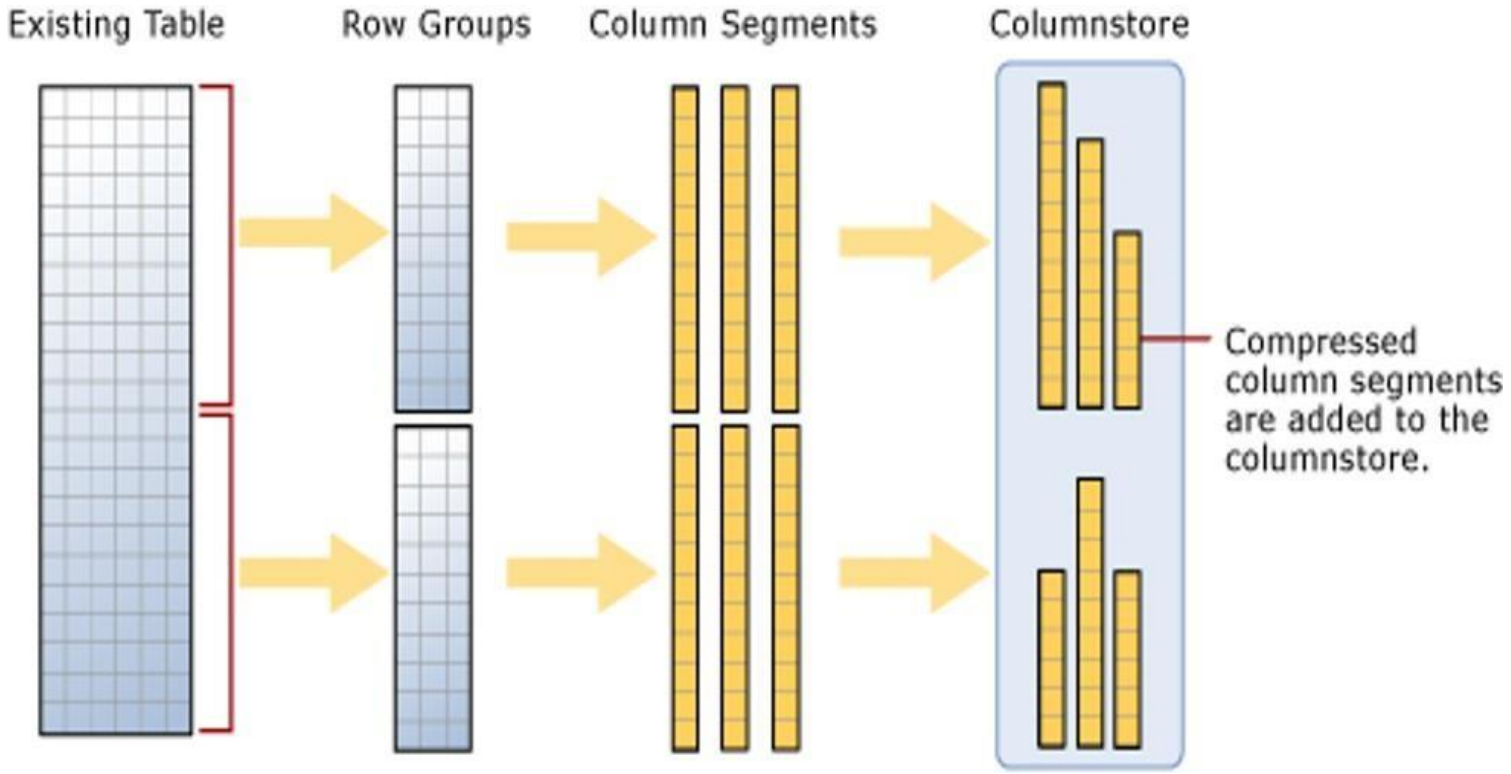
Non-clustered index

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

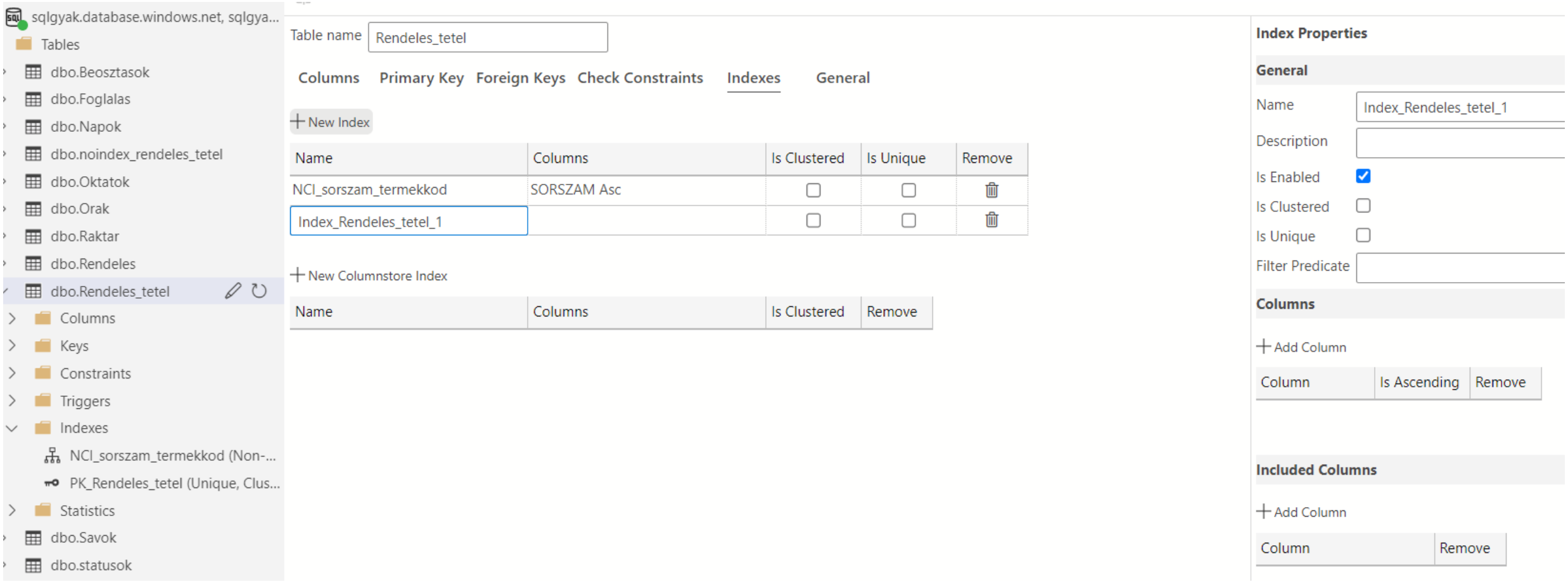
A columnstore index jellemzője az oszlop-alapú tárolás és lekérdezés végrehajtás



- 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…

…

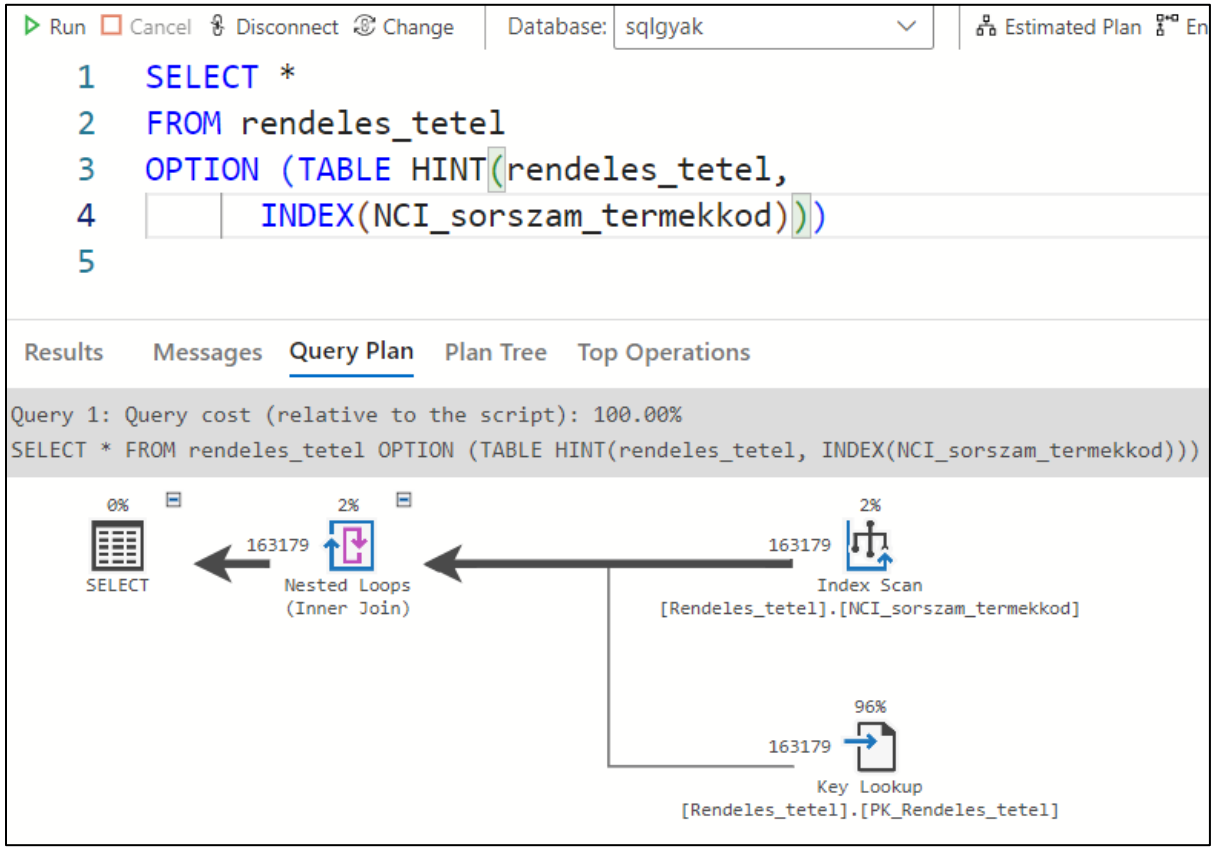
ORDER BY …

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

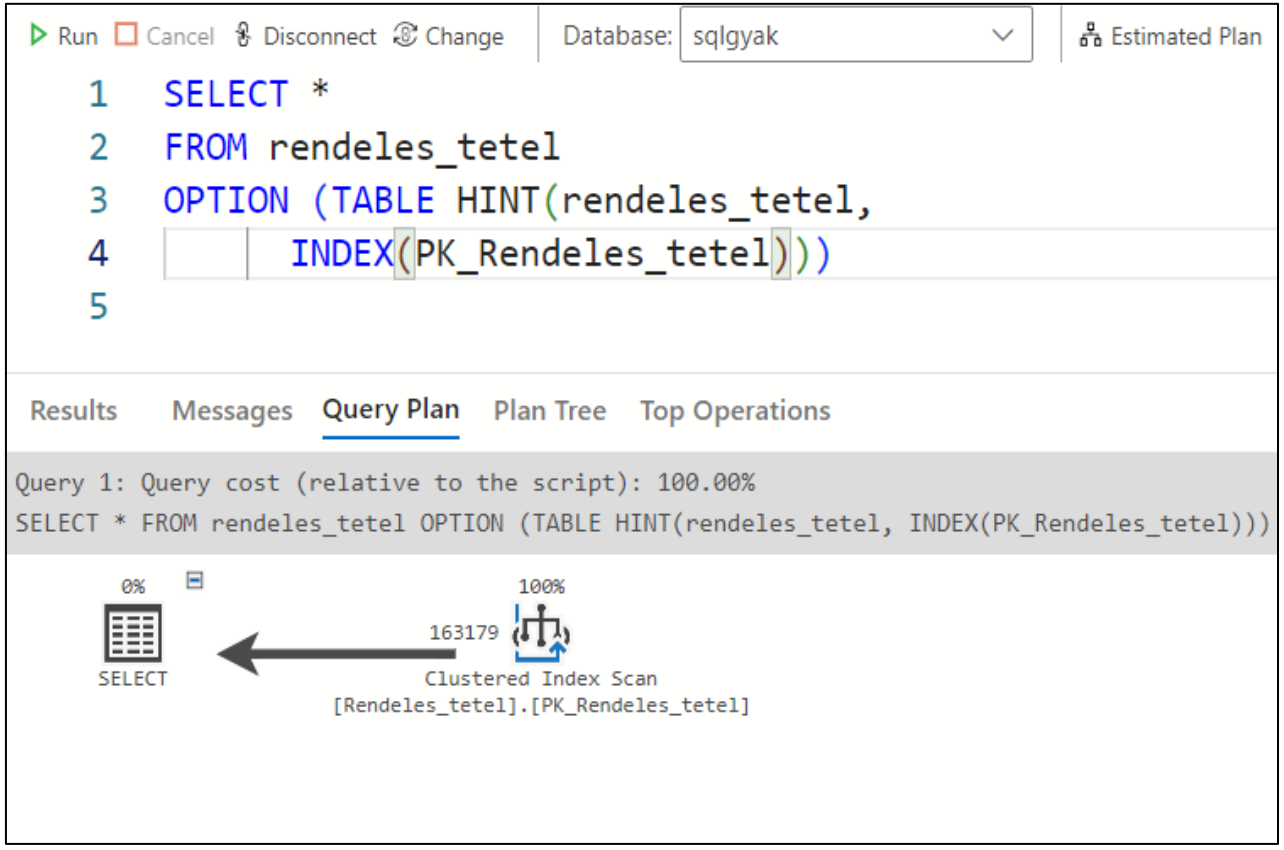
- Többféle HINT létezik:
  - JOIN HINT
  - TABLE HINT
  - QUERY HINT
- Céljuk:
  - Preferált vagy tiltott index
  - JOIN típus, sorrend
  - Táblaelérési mód
  - Párhuzamos végrehajtás

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?

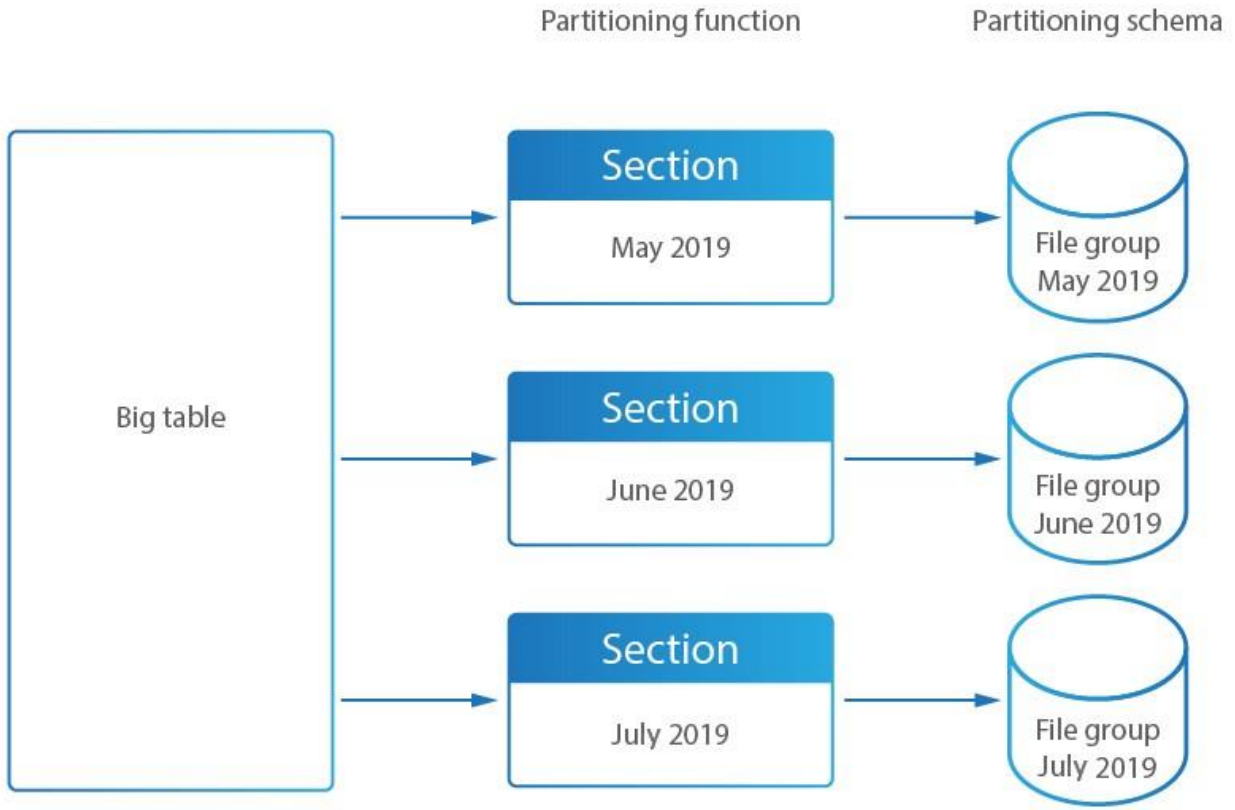
.Az adatbázis tartalma egy vagy több filegroup-ban (fájlcsoportok) tárolódik. Egy fájlcsoport tartalmazhat elsődleges és másodlagos adatfájlt és log fájlt.

- **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.

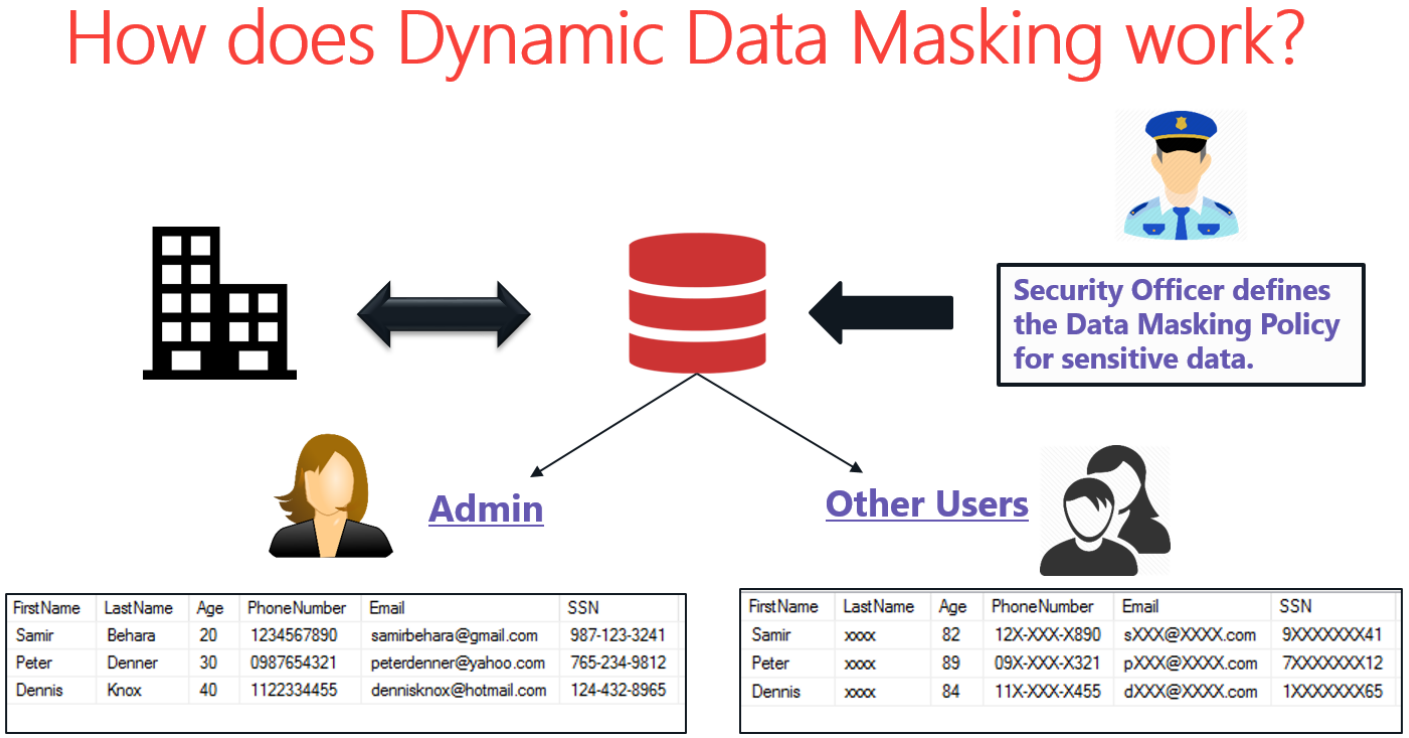
- A lapok rendszerint valamilyen stuktúrába vannak szervezve, pl. halom, B-fa
- A lapoknál nagyobb tárolási egység az extent, amely 8 db egymás melletti lapot jelent
- Hasonló tárolási egység a block, amely fájlrendszertől függően adott menyiségű lapot vagy extent-et jelent

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)



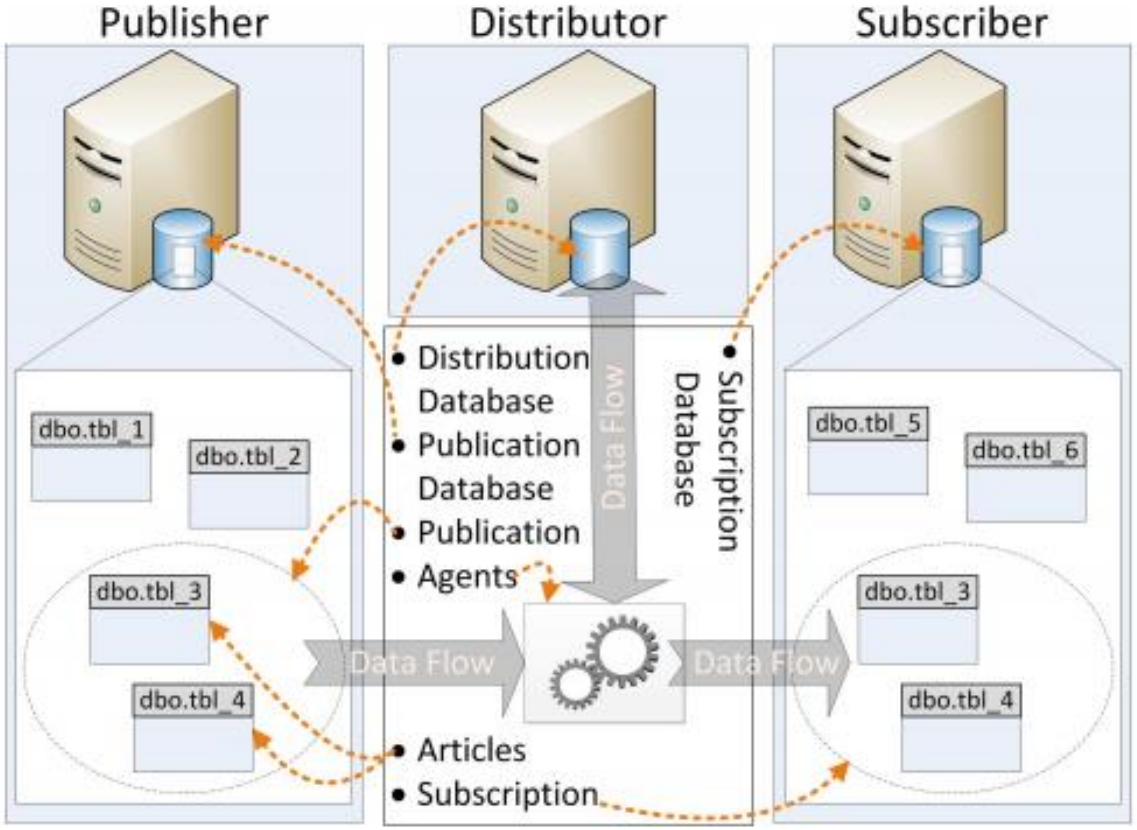
Forrás: Dynamic Data Masking – Altering the masked column – samirbehara

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:

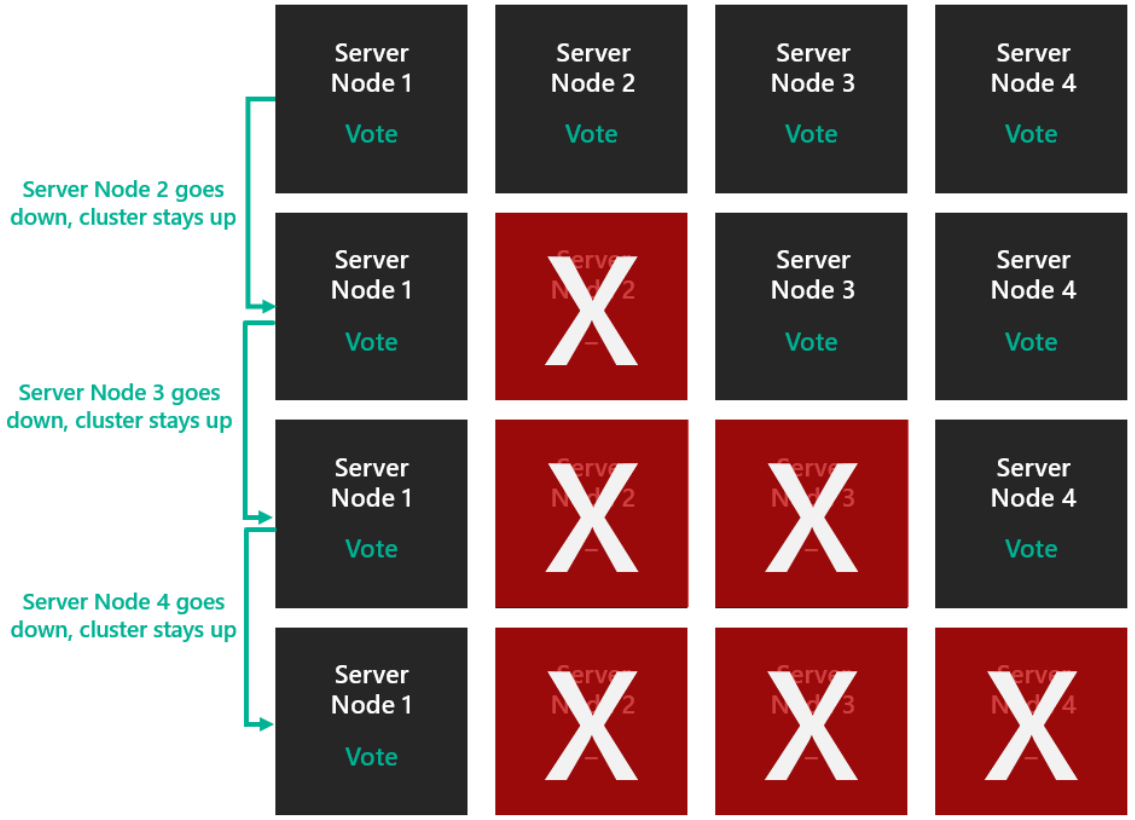
- Snapshot replikáció
- Tranzakciós replikáció
- Peer-to-peer replikáció



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!