A black and blue text

Description automatically generated

**Adatbázisokelőadás05**

A group of circles in a circle

Description automatically generated

1

# Fejlesztési és konfigurálási feladatok

Adatok

bevitele

Nézetek

Tárolt

eljárások

Függvények

Triggerek

Tranzakciók

Jogosultságok

Indexek

Adattárolás

A black and blue text

Description automatically generated

**Indexek**

**Hogyanlehetgyorsítanialekérdezéseksebességét?**

A screenshot of a computer

Description automatically generatedSok 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
* Alogikai 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 | |  |  |  |  | | --- | --- | --- | --- | |  | **D** | **OLGOZÓ** | | | 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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A yellow rectangular sign with black text  Description automatically generated | | |  | | --- | | Az indexmutató egy rekordra mutat. | | |
| A yellow rectangular sign with black text  Description automatically generated | | |  | | --- | | Az indexmutató egy blokkra mutat. Ablokkon 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.Amutató a rekord fizikai helyére mutat.

A diagram of a person with lines

Description automatically generated with medium confidence

# Többszintes indexek

A yellow rectangular objects on a black background

Description automatically generatedAz 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

A diagram of a network

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

# B-fák tulajdonságai

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

# B-fák (kiegyensúlyozott keresőfák)

A diagram of a company

Description automatically generated

# Hash-alapú indexek

* A diagram of a function

  Description automatically generatedAz adatok csoportokba vannak rendezve
* A hash függvény adja meg, hogy melyik csoportban van az adat

# Bitmap indexek

* A screenshot of a computer code

  Description automatically generatedOlyan oszlopokra alkalmazzuk, ahol kevés az egyedi érték
* Abitmap 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.

* Aclustered 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

Anon-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

|  |
| --- |
| Adattárházakból való lekérdezéseknél kiemelten fontos! |

* A diagram of a graph

  Description automatically generatedAcolumnstore index lehet
* Clustered
* Non-clustered
* Egy táblához csak egy columnstore index készíthető

# Indexek – Azure Data Studio

A screenshot of a computer

Description automatically generated

**Melyik index legyen használva?**

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

HINT- Lekérdezési tipp

Használata:

SELECT ….

FROM…

…

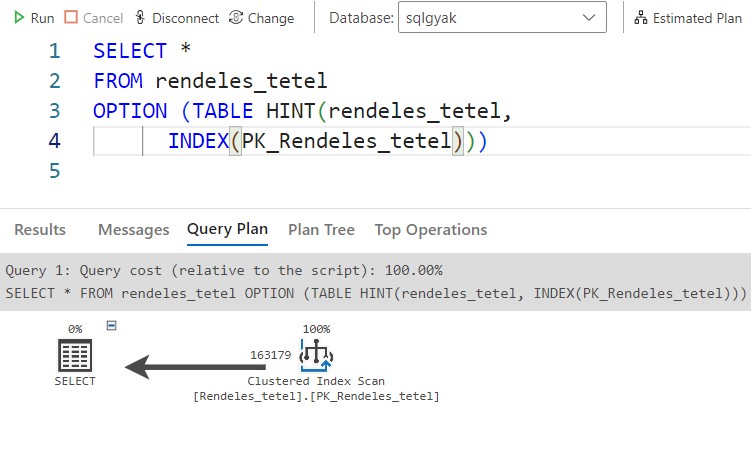
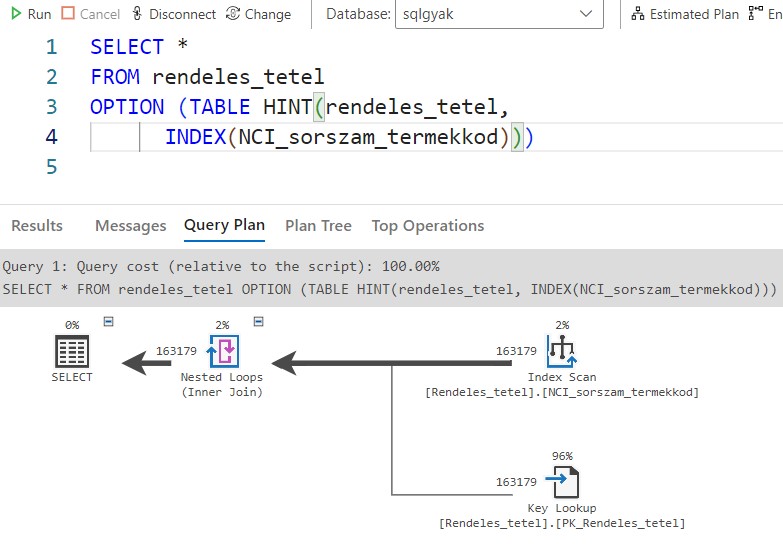
ORDER BY …

**OPTION**

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

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

# Table HINT példa – index specifikálás



A lekérdezés futtatása

Clustered

index segítségével

A lekérdezés futtatása egy saját index

segítségével

# Indexek – hasznos parancsok

A screenshot of a computer

Description automatically generated

**Az indexek hátrányai**

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

# Adattárolás

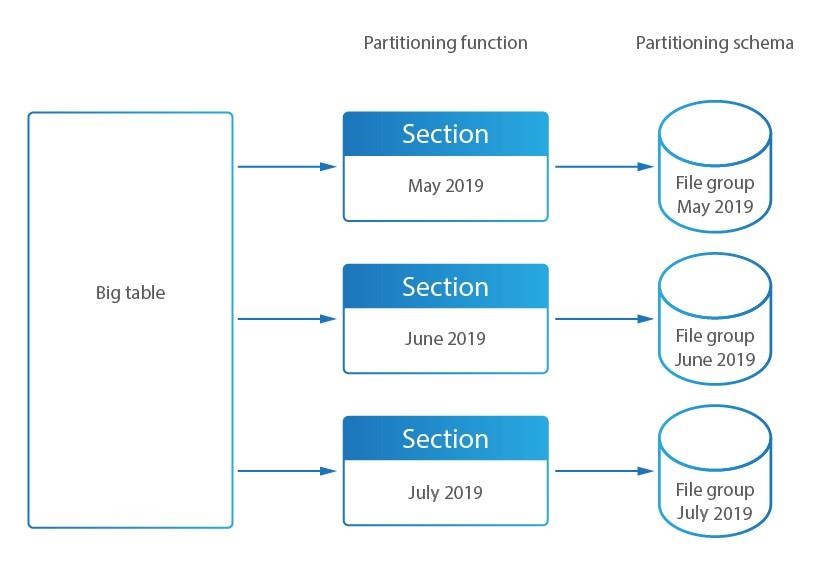
**Hogyan tárolódik fizikailag egy adatbázis?**

.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.

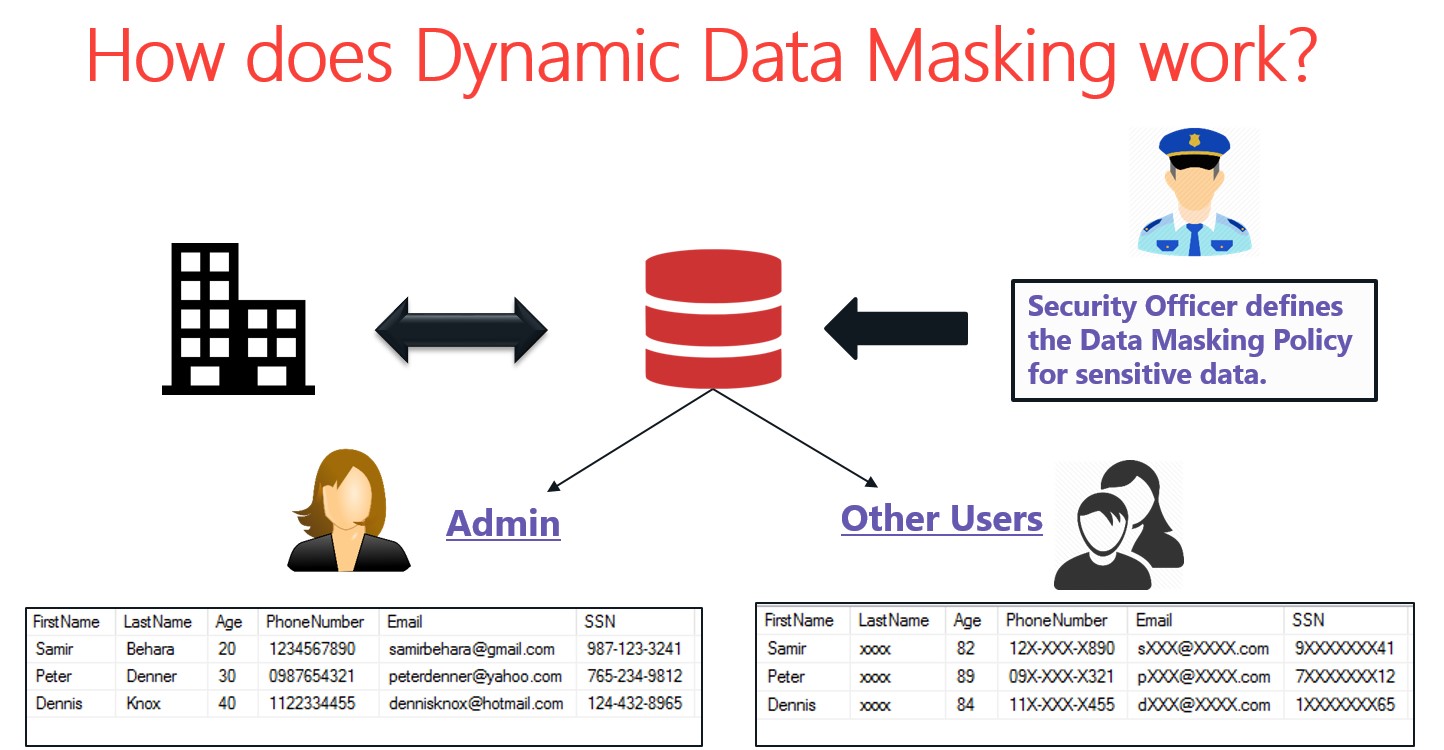
* **Sorszintű tömörítés**. A redundáns információkat csak egyszer tárolja le. Különösen hatékony, ha a tábla sok redundanciát tartalmaz.
* **Lapszintű tömörítés**. Az egész lapot tömöríti egy adott tömörítési algoritmus alapján. Legtöbbször hatékonyabb, mint a sorszintű tömörítés.

\*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 jelent31

# 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

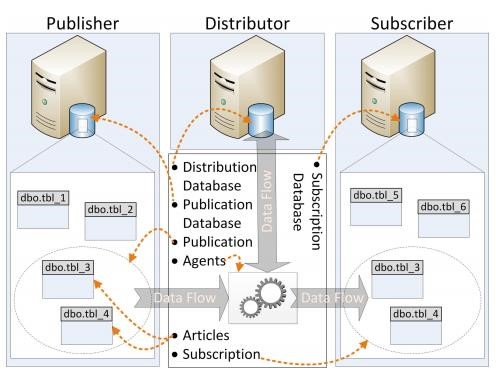
(kulcsot + konfigurációt igényel)

* Transparent Data Encryption

(csak Enterprise verziónál) [Forrás:](https://samirbehara.com/2017/10/28/dynamic-data-masking-altering-the-masked-column/) [Dynamic Data Masking –](https://samirbehara.com/2017/10/28/dynamic-data-masking-altering-the-masked-column/) [Altering the masked column](https://samirbehara.com/2017/10/28/dynamic-data-masking-altering-the-masked-column/) [–](https://samirbehara.com/2017/10/28/dynamic-data-masking-altering-the-masked-column/) [samirbehara](https://samirbehara.com/2017/10/28/dynamic-data-masking-altering-the-masked-column/)

# Replikáció

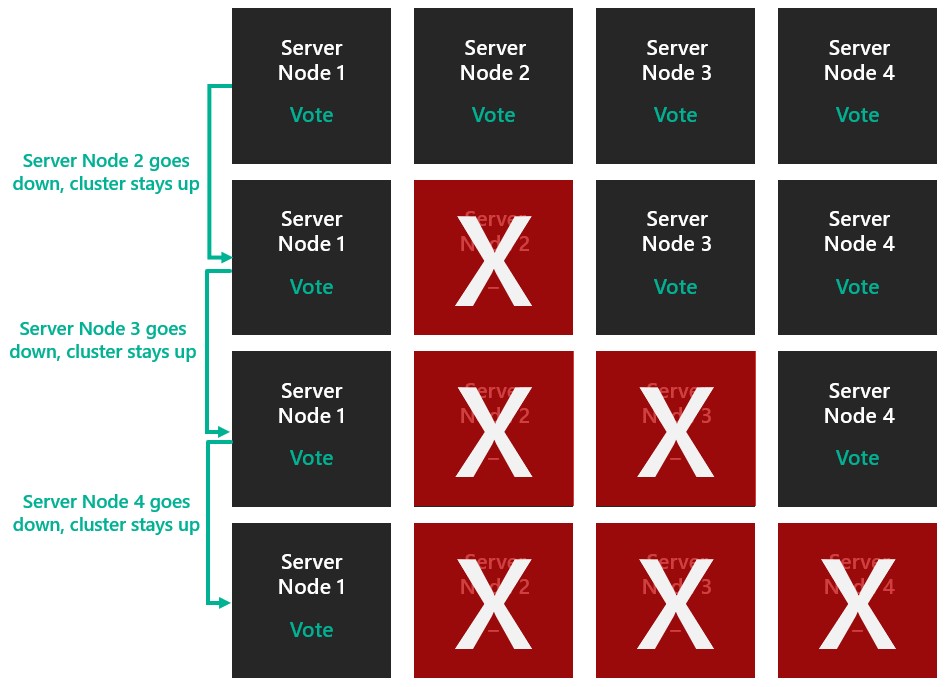
Egy SQL szerver példányon történő adatok redundáns tárolását, és annak rendszeres szinkronizálását jelenti egy vagy több másik SQL szerver példányon

A replikáció többféle módon megvalósulhat, pl:

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

# Klaszter

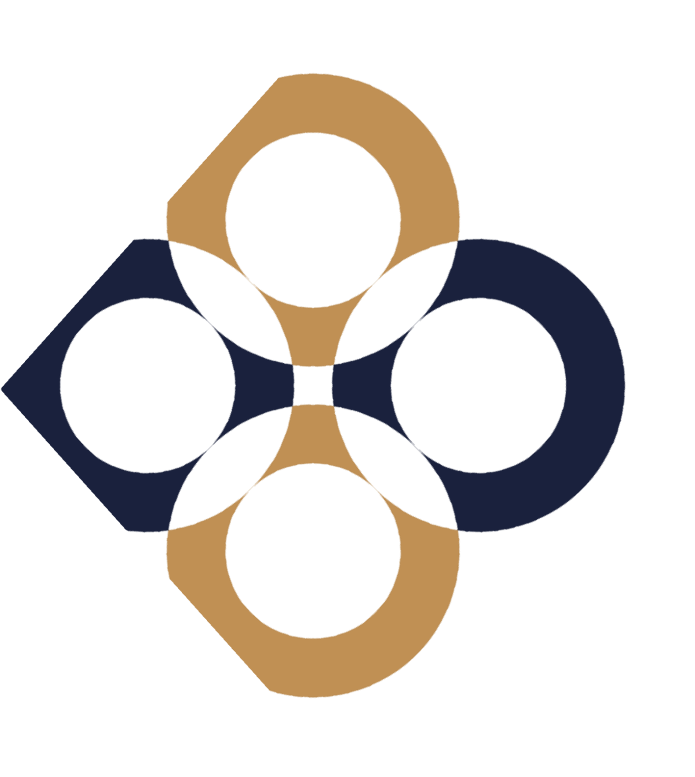
SQL szerverek csoportja, amelyek együttműködése lehetővé teszi a magas rendelkezésre állás és a hibatűrés megvalósítását

* 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 -](https://learn.microsoft.com/en-us/azure-stack/hci/concepts/quorum) [Azure Stack HCI | Microsoft Learn](https://learn.microsoft.com/en-us/azure-stack/hci/concepts/quorum)

**Adatbázisok előadás 06**

Adattárházak 

**Miről lesz szó?**

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

# Történetiáttekintés - címszavakban



# Vezetőiinformációs 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 máselnevezés**

Adattárház

Döntés

támogató

rendszer

Vezetői

információs

rendszer

Üzleti

intelligencia

rendszer

Analitikus

rendszer

Tudásbázis

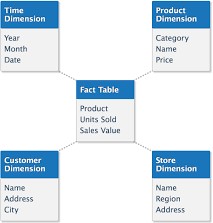
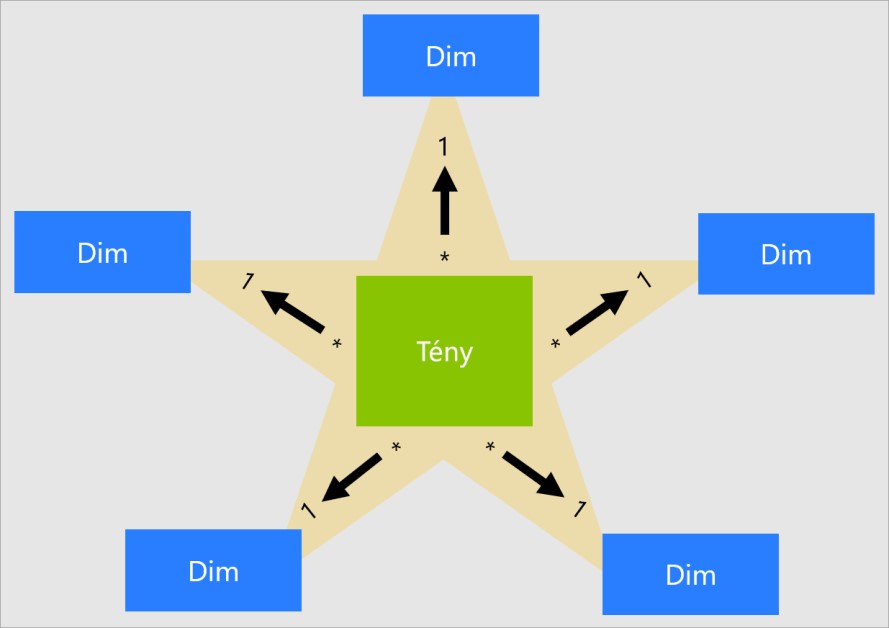
Adatbank

# (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



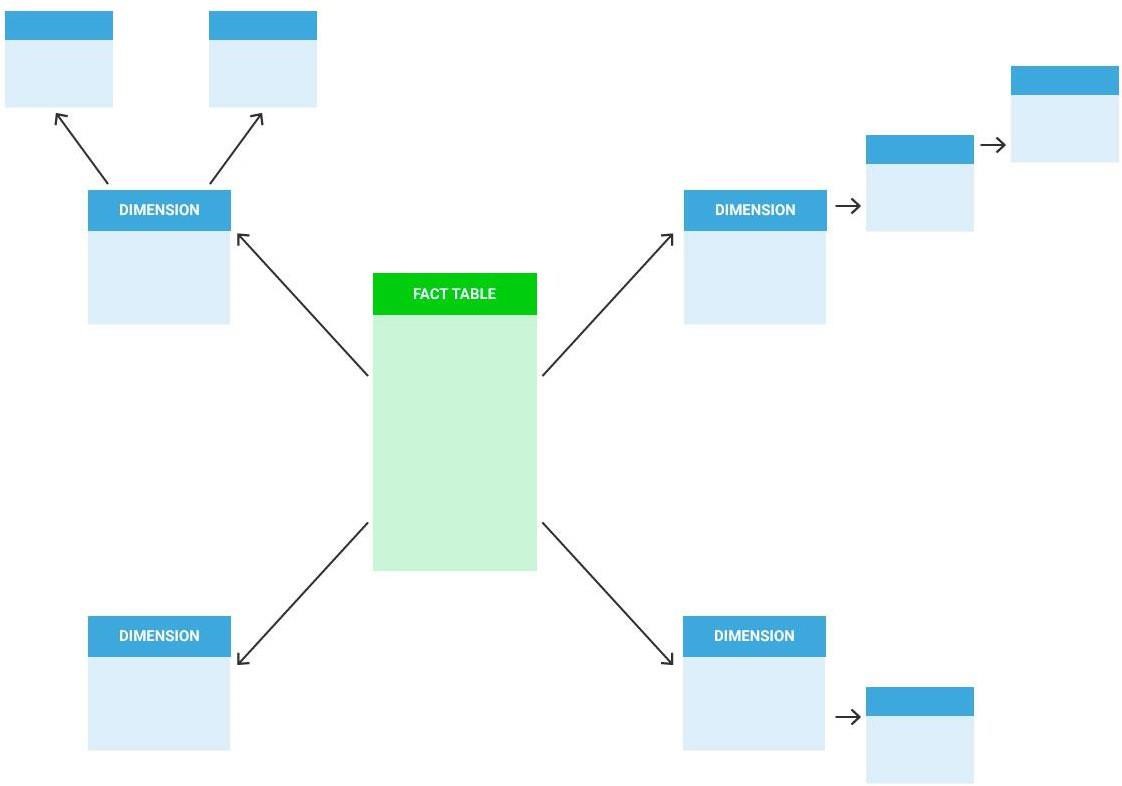
## Tényadatok(ténytáblák)vs.Dimenzionális adatok (dimenziók)

|  |  |  |
| --- | --- | --- |
| **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

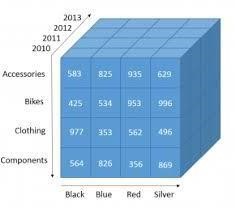
# Hópehely (Snowflake) séma

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

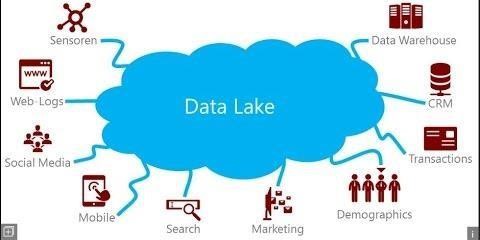
|  |  |  |
| --- | --- | --- |
| Az adatpiac nem normalizált, aggregált adatokat tartalmaz, és valamely szervezeti egység működési, felhasználói követelményei által meghatározott (Inmon, 2002) |  | Az adatpiac az adattárház egy tárgyterülethez (pl. marketing) köthető része (Turban, 2014) |

Az adatpiacok az adattárházakhoz hasonló adatkezelési képességekkel rendelkeznek, de egyegy 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

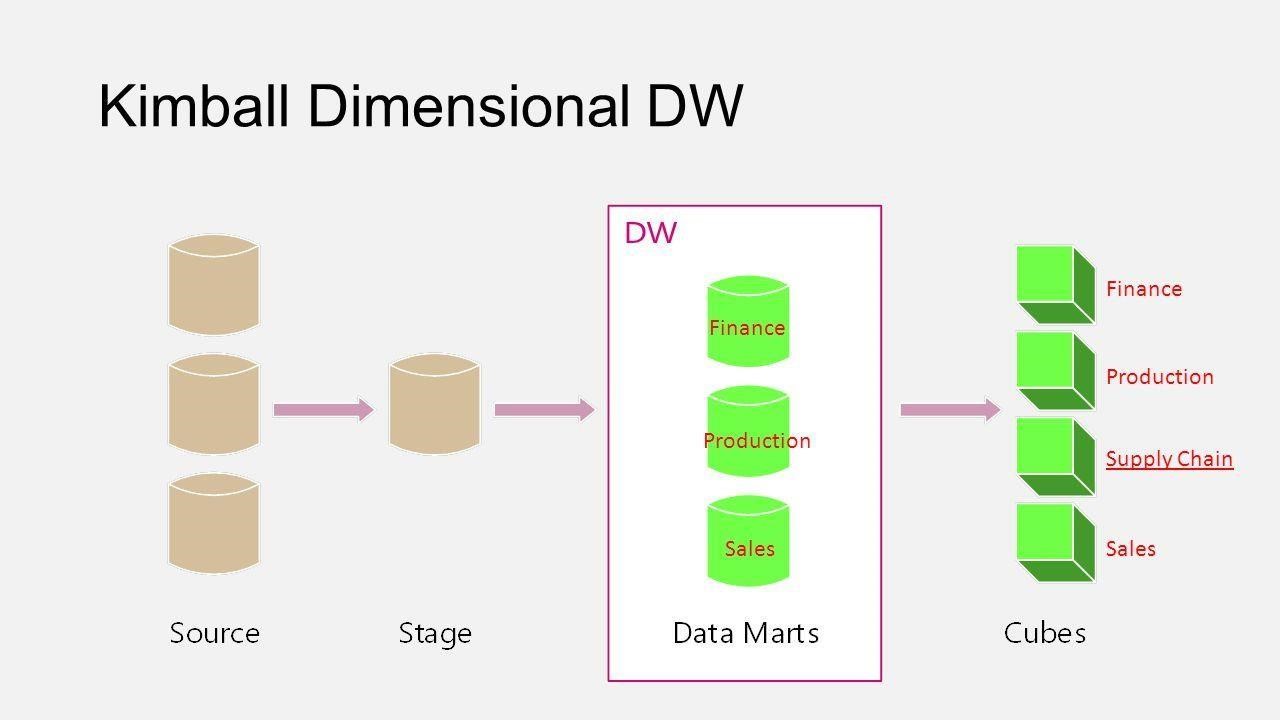
* Big Data rendszerekre jellemző
* Központosított adatkezelés
* Az adattárházaknál rugalmasabb struktúra ❑ Strukturált és nem strukturált adatokat is tartalmazhatnak

A kép forrása: https://[www.youtube.com/watch?v=a3nezIjxI-E](http://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 |

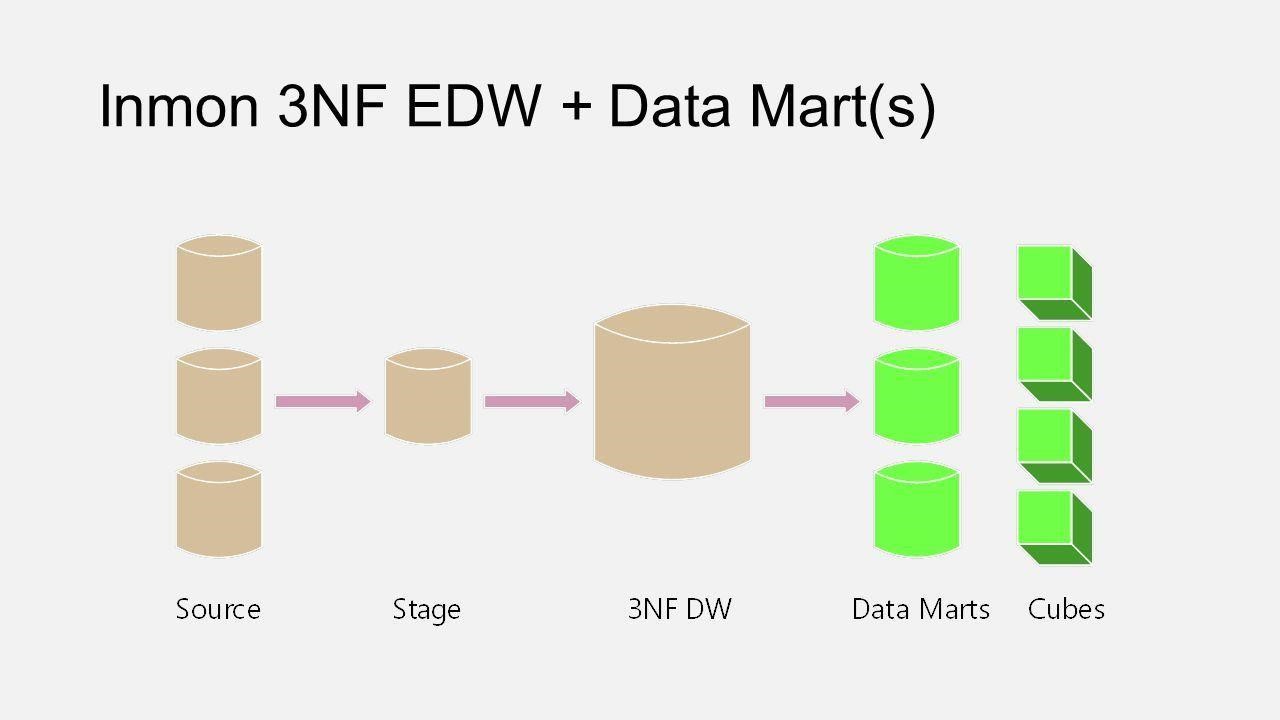
**Kimball adattárház modellje – alulról fel megközelítés**



**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 adattárházmodellje – fentről le megközelítés



**Inmon adattárházmodellje – 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

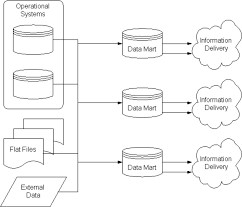
* a felhasználók IT területről kerülnek ki
* inkább taktikai döntések szükségesek
* a forrásrendszerek viszonylag stabilak
* minél előbbi eredményt szeretnénk elérni, kis kezdeti befektetéssel és csapattal
* az adatok különálló üzleti területekről jönnek
* a változások köre limitált

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**

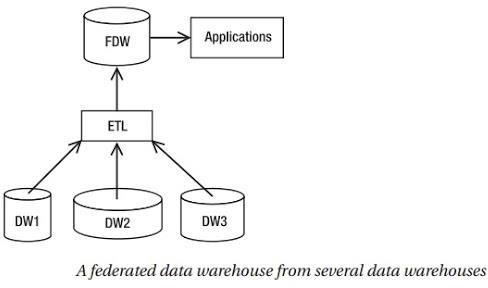
**Példa1: független adatpiacok(independentdatamarts)**

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

[A](https://www.researchgate.net/figure/Data-flow-when-using-independent-data-marts_fig6_34267502) [kép](https://www.researchgate.net/figure/Data-flow-when-using-independent-data-marts_fig6_34267502) [forrása:](https://www.researchgate.net/figure/Data-flow-when-using-independent-data-marts_fig6_34267502) [https://www.researchgate.net/figure/Dataflow-when-using-independent-data-marts\_fig6\_34267502](https://www.researchgate.net/figure/Data-flow-when-using-independent-data-marts_fig6_34267502)

**Alternatívadattárházarchitektúrák**

**Példa 2:egyesített adattárház(federateddatawarehouse)**

Több adattárházat egyesít

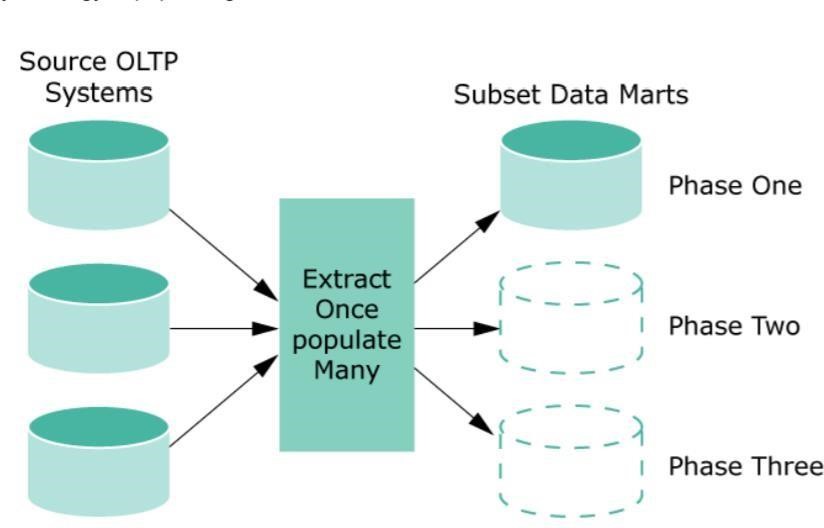
* Pl: nagyobb cég, telephelyek több országban, mindegyiknek saját adattárház
* Közös üzleti szabályok, egyszerű lekérdezés
* Nehéz technikai megvalósítás

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

# Alternatívadattárházarchitektúrák

## Példa 3:Inkrementálisan felépített adatpiacok

**(incremental architected datamarts)**

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

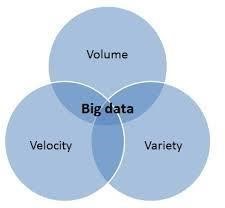
A kép forrása:

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

**Adattárházak – új kihívások**

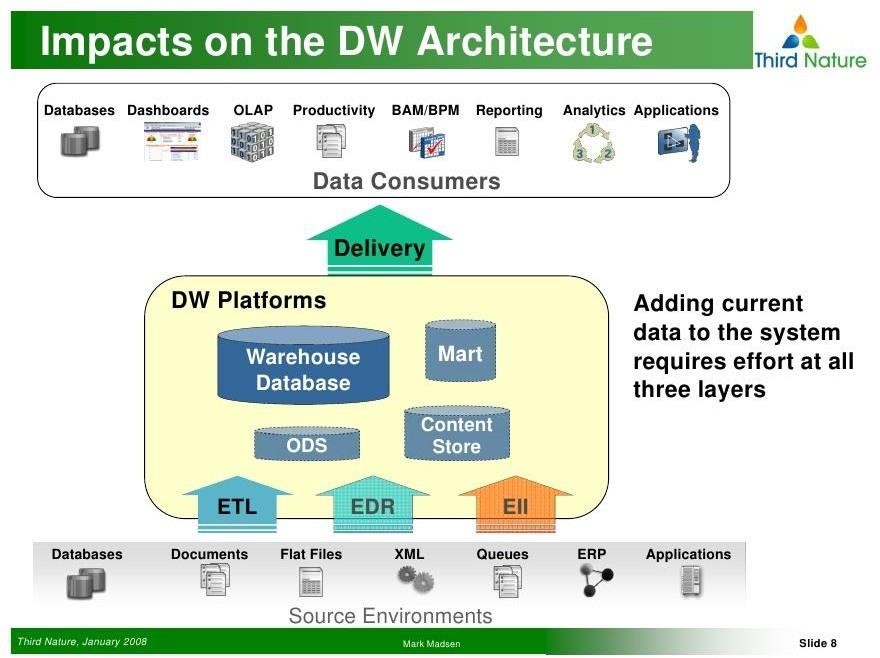
* Növekvő adatmennyiség
* Adatfrissítés szinte azonnal
* Gyakori változások az adatmodellben
* Lekérdezések futtatása a memóriában

**Egyre növekvőadatmennyiség - 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

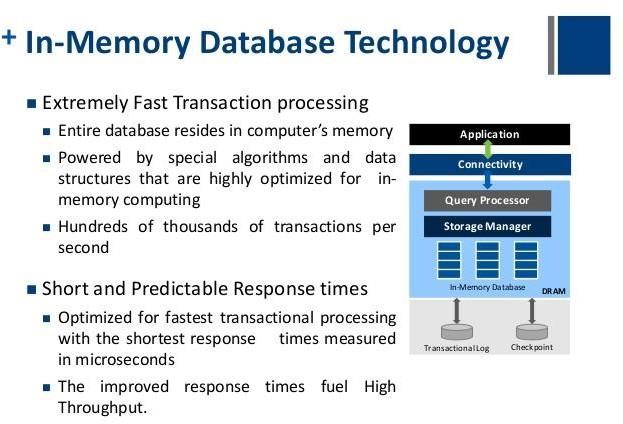
A kép forrása: <https://www.slideshare.net/mrm0/how-real-time-data-changes-the-data-warehouse>

# 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

A kép forrása:

[https://www.slideshare.net/A ltibase/solve-big-dataproblem-the-inmemorydatabase-solution17179969](https://www.slideshare.net/Altibase/solve-big-data-problem-the-in-memorydatabase-solution-17179969)

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

**Adatbázisok előadás 04**

Fizikai adatbázisterv

Adatbázisok fejlesztése A group of circles in a circle

Description automatically generated

# Atervezés lépései

Fogalmi Logikai Fizikai adatmodell adatmodell adatmodell

Egyed – kapcsolat

modell (ER-modell)Adatbázisterv**Implementálás terve**

**Logikai adatmodell**

## Logikai adatmodell\* (Dvd adatbázis)

A diagram of a computer server

Description automatically generated\* 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.)

## CASE-eszköz

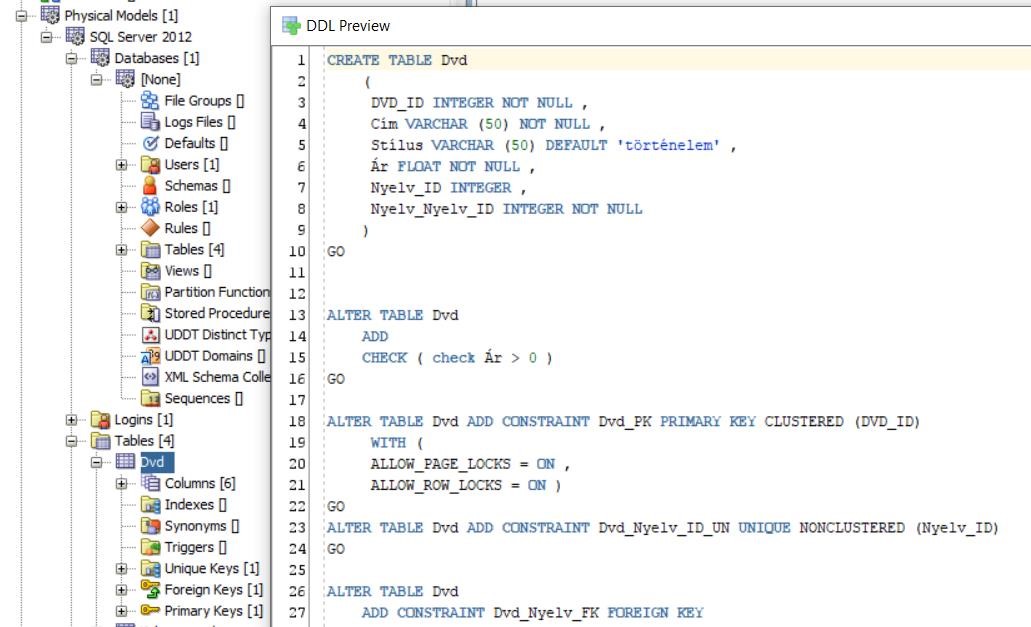
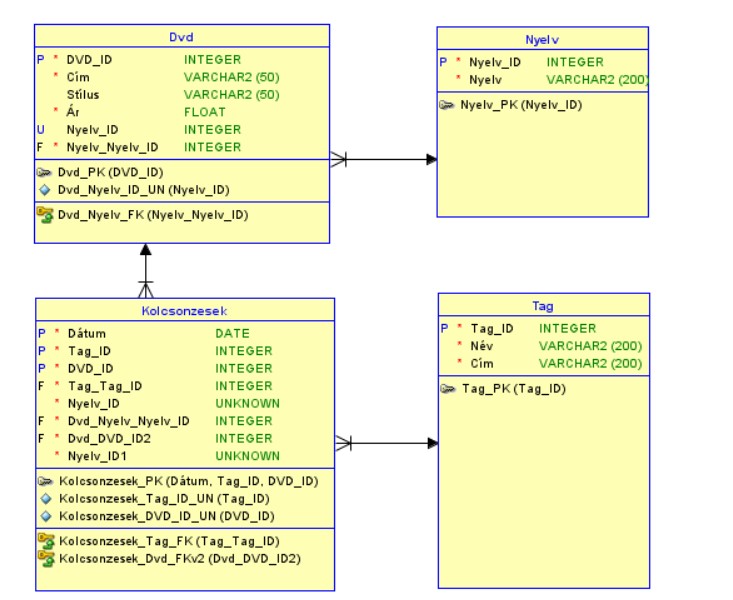
Adatmodellezés esetén olyan szoftver, amely segíti a modellezés teljes folyamatát

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

Adatok

bevitele

Nézetek

Tárolt

eljárások

Függvények

Triggerek

Tranzakciók

Jogosultságok

Indexek

Adattárolás

**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)

Anézet egy elmentett, névvel ellátott lekérdezés.

* Anézetekből 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

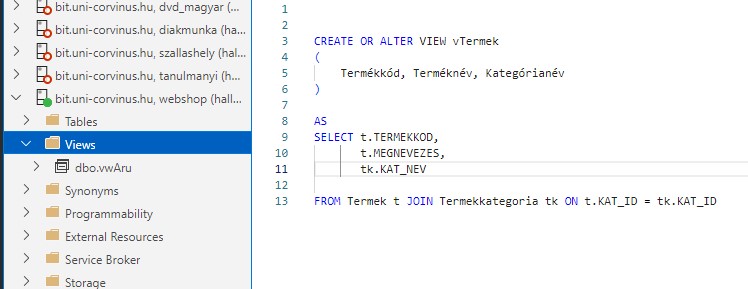
|  |  |  |
| --- | --- | --- |
| Korlátozható az adatok elérése |  | A bonyolultabb lekérdezések egyszerűbb formára hozhatók |
| Az adatokat többféle  nézőpontból szemlélhetjük |  | Az adatfüggetlenség biztosítása |

## 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**

Tko

d

Tnev

Tszulido

T01

Kiss

Béla

1999.01.01

T02

Nagy

Ilona

2003.02.12

**OSZTÁLYZAT**

Tkod

Tankód

Jegy

T01

Tan01

5

T01

Tan02

3

T02

Tan01

4



**TANTARGY**

Tankod

Tannév

Tan01

Algebra

Tan02

Analízis

Tan03

Programozás



|  |  |  |
| --- | --- | --- |
| **V\_OSZTALYZAT** | | |
| Tnév | Tannév | Jegy |
| Kiss Béla | Algebra | 5 |
| Kiss Béla | Analízis | 3 |
| Nagy Ilona | Algebra | 4 |

CREATE VIEW V\_OSZTALYZAT AS

SELECT t.tnev AS 'TNév’, tt.tannev AS 'Tannév’, o.jegy

FROM Osztalyzat o

JOIN Tanulo t ON o.tkod = t.tkod

JOIN Tantargy tt ON o.tankod = tt.tankod

**Tárolt eljárások**

## SQL kód vs. Alkalmazások

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



Tesztelés?

Karbantartás?

Server

-

kliens

kapcsolatok

száma?

SQL

kód

újra

felhasználása?

Jogosultságok?

## A tárolt eljárások működése

Diagram of a diagram of a cloud

Description automatically generated

Akép forrása: Bina Nusantara University

## Tárolt eljárás (Stored procedure)

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

Atárolt 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 | * Akódok 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

|  |  |  |
| --- | --- | --- |
| A screenshot of a computer  Description automatically generated | |  | | --- | | CREATE PROCEDURE procedure\_name  -- paraméterek megadása  AS  BEGIN  -- SQL utasítások  END | |

## 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

Afüggvények 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

|  |  |  |
| --- | --- | --- |
| A screenshot of a computer  Description automatically generated | |  | | --- | | CREATE FUNCTION function\_name  (  -- paraméterek megadása  )  RETURNS adattípus  AS  BEGIN  -- SQL utasítások  RETURN érték  END | |

**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

**DMLTriggereklétrehozásaMSSQL-ben**

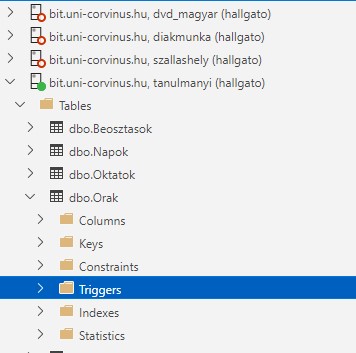
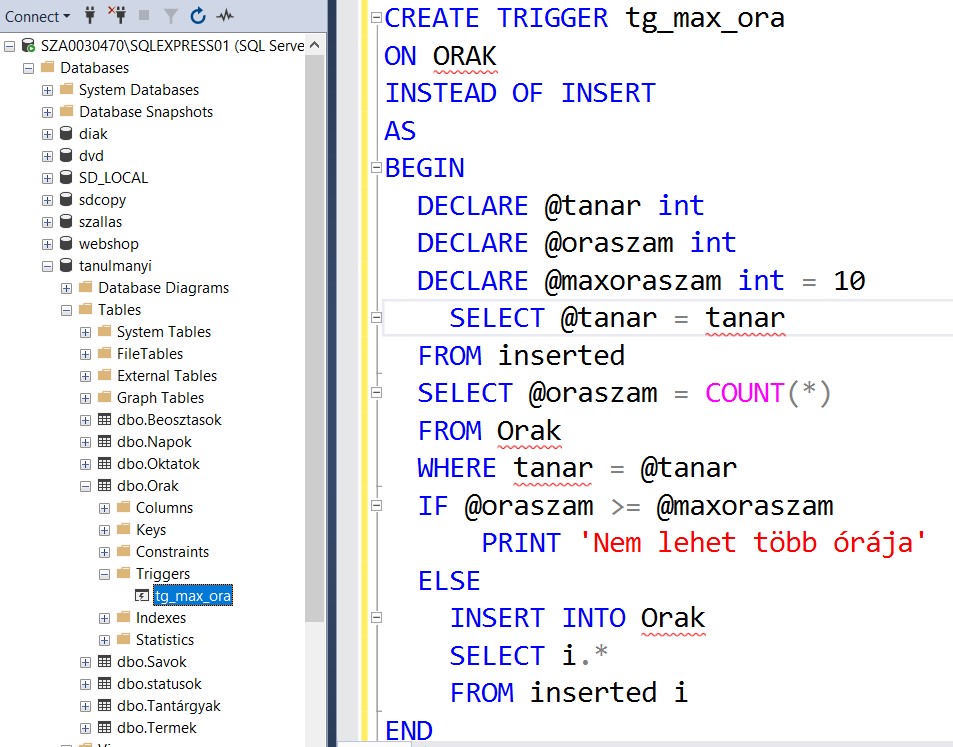
|  |
| --- |
| CREATE TRIGGER triggernév  ON táblanév  FOR | AFTER | INSTEAD OF INSERT | UPDATE | DELETE  AS  BEGIN  -- SQL utasítások  END |

Pl: egy oktatónak maximum

10 órája lehet

## Triggerek - példa

Az inserted tábla tartalmazza az új vagy módosult sorok másolatát



### Triggerek –előnyökéshátrányok

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | * 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 | |

**Tranzakciók**

## - átutalás

Számla1

Egyenleg:

Változás:

250

000

Ft

-

50

000

Ft

Új

egyenleg:

200

000

Ft

Számla2

Egyenleg:

Változás:

Új

egyenleg:

300

000

Ft

+50

000

Ft

350

000

Ft



Átutalás:

50

000

Ft

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?

## – átutalás (folytatás)

Az átutalás folyamata megszakadhat pl. az első lépés után

Számla1

Egyenleg:

Változás:

250

000

Ft

-

50

000

Ft

Új

egyenleg:

200

000

Ft

Számla2

Egyenleg:

Változás:

Új

egyenleg:

300

000

Ft

+0

Ft

300

000

Ft



Átutalás:

50

000

Ft

A folyamat megszakadása esetén inkonzisztens adatok lehetnek az adatbázisban

## – átutalás (megoldás)

Kezeljük egyetlen logikai egységként az átutalás 2 fő lépését!

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Számla1  Egyenleg: 250 000 Ft Változás: - 50 000 Ft  Új egyenleg:  **Átutalás: 50000Ft**  Számla2  Egyenleg: 300 000 Ft Változás: +50 000 Ft  Új egyenleg: | | |  | | --- | | Számla1  Egyenleg: 250 000 Ft Változás: - 50 000 Ft Új egyenleg: 200 000 Ft  **Átutalás: 50000Ft** | | 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 nem lép fel hiba, akkor az összetett

lépéssorozat rendben végrehajtódik Ha valahol megszakad a folyamat, akkor visszaáll az eredeti állapot

## Probléma2 – átutalás

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | Számla1  Egyenleg: 250 000 Ft | |  | |  | | --- | | Számla1  **Átutalás: 50000Ft** Számla2 | |
| |  | | --- | | Számla1  **Átutalás: 100000Ft** Számla2 | |
| |  | | --- | | Számla2  Egyenleg: 300 000 Ft | |
| |  | | --- | | Számla3  **Átutalás: 200000Ft** Számla2 | |

38

## Probléma2 – átutalás (megoldás)

Korlátozni (szabályozni) kell az egyidejű hozzáférést!

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  | | --- | | Számla1  Egyenleg: 250 000 Ft | |  | |  | | --- | | Számla1  **Átutalás: 50000Ft** Számla2 | |
| |  | | --- | | Számla1  **Átutalás: 100000Ft** Számla2 | |
| |  | | --- | | Számla2  Egyenleg: 300 000 Ft | |
| |  | | --- | | Számla3  **Átutalás: 200000Ft** Számla2 | |

## Tranzakció

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

A hand inserting a paper into a machine

Description automatically generatedAtranzakció 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/10

## Tranzakció tulajdonságok - ACID

A colorful rectangular signs with white text

Description automatically generated with medium confidence**Atomicity**

Nem valósulhat meg részlegesen

**Consistency**

Végrehajtása után az állapot konzisztens marad (pl. kényszerek teljesülnek)

**Isolation**

A párhuzamosan futó tranzakciók nem zavarhatják egymást

A kép forrása: https://dev.to/princessanjana1996/acid-

properties-in-databases-43aa **Durability**

Sikereslefutásután a változás tartósan megmarad

## Zárolás (lock) fogalma

A zárolás olyan eszköz, amely segítségével az adatbáziskezelő rendszer korlátozza az adatok egyidejű elérését a tranzakciók számára.

* Azárolásnak 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 |

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

Zárolási módok

## Zárolási kompatibilitás az MS SQL-ben

A screenshot of a computer

Description automatically generated

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 | | Tranzakció1  Olvas:  DB=350  Eladás:  10  DB  Módosítás:  DB=340 | |  | | --- | | Tranzakció2  Olvas: DB=350  Eladás: 20 DB  Módosítás: DB=330 | |

### Egyidejűtranzakciók kezelése

„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 | | |  | | --- | | Tranzakció1  Olvas: DB=350  Eladás: 10 DB  Módosítás: DB=340  Hiba miatt vissza:  DB = 350 | |  | |  | | --- | | Tranzakció2 Olvas: DB=340 | |

### Egyidejűtranzakciókkezelése

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 | | |  | | --- | | Tranzakció1  Olvas: DB=350  Olvas: DB = 340 | |  | |  | | --- | | Tranzakció2  Eladás: 10 DB  DB=340 | |

### Egyidejűtranzakciókkezelése

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

T

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  | **KÉSZLET** |  | | ID | Termék | DB | | 1 | Tégla | 350 | | 2 | Cement 50 kg | 50 | | |  | | --- | | Tranzakció1  INSERT:  (3, ’Festék’, ’12’) | |  | |  | | --- | | Tranzakció2  Olvas:  SELECT COUNT(\*) FROM KÉSZLET  Olvas:  SELECT COUNT(\*)  FROM KÉSZLET | |

# 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.

## Konkurencia problémákés 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 @@TRANCOUNTértéke 0
* Minden BEGIN TRANSACTION 1-gyel növeli, minden COMMIT1-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 aktiváltható

## Tranzakciót kiváltó SQL-utasítások

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

**Explicit tranzakciókmegvalósítása 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

**Implicittranzakciók megvalósításaSQL-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;

(UPDATEjog adása Tedfelhasználó számára a Product táblához)

## Jogosultságok (SQL-szerver)

A screenshot of a computer

Description automatically generatedAz 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** | **Rövidleírás** |
| 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 |

## Adatbázis-szintű szerepkörök

A screenshot of a computer

Description automatically generated

**Táblajellegűobjektumokjellemző jogosultságai**

### (táblák,nézetek,tábla-értékűfüggvények)

|  |  |
| --- | --- |
| **Jogosultság** | **Elvégezhető művelet (korlátozható oszlopokrais)** |
| **SELECT** | olvasás táblázatból,nézetből |
| **INSERT** | adatbevitel táblázatba, nézetbe |
| **DELETE** | Sor(ok) törlése táblázatból, nézetből |
| **UPDATE** | adatok módosítása táblázatban,nézetben |
| **REFERENCES** | idegen kulccsal való hivatkozástáblázatra |
| **ALL** | minden művelet |

## Jogosultság-hierarchia A screenshot of a computer Description automatically generated

**Jogosultságoklekérdezése– 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

A screenshot of a computer

Description automatically generated

**Jogosultságoklekérdezése–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;

A screenshot of a computer

Description automatically generated

**Köszönöm afigyelmet!**