RELÁCIÓTÍPUSOK SQL-BEN

ADATBÁZISOK

Dóka - Molnár Andrea

andrea.molnar@math.ubbcluj.ro

BABEŞ-BOLYAI TUDOMÁNYEGYETEM

Matematika és Informatika Kar

Visszacsatolás: Alkérdések a FROM záradékban

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

- Oldjuk meg a HAVING záradék kiküszöbölésével!

(1) Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500 €!

Visszacsatolás: Alkérdések a FROM záradékban

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

- Oldjuk meg a HAVING záradék kiküszöbölésével!

(1) Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500 €!

```
SELECT ReszlegID, Atlag
FROM (SELECT ReszlegID,AVG(Fizetes)[AS] Atlag
        FROM Alkalmazottak
       GROUP BY ReszlegID) [AS] ReszlegAtlag
WHERE Atlag > 500
```

- Itt: ReszlegAtlag(Atlag) ~ **NÉZET**

Relációtípusok SQL-ben

Reláció ↔ tábla (TABLE) (SQL-beli megnevezés)

3 típusú tábla SQL-ben:

1. **(Alap)tábla/tárolt reláció (table)**
2. **Ideiglenes/temporális tábla (temporary table)**
3. **Nézet(tábla) (view)**

Relációtípusok SQL-ben

1. **(Alap)tábla/tárolt reláció (table):**
   - A tárolás alapegysége; sorok halmaza.
   - Fizikailag léteznek az adatbázisban ↔ az adatbázisrendszer valamilyen fizikai struktúrában tárolja őket.
   - Nem változnak addig, amíg valamilyen táblamódosító SQL-utasítás meg nem változtatja őket.
   - DML műveletek végrehajtása – minden esetben.

Relációtípusok SQL-ben

2. Ideiglenes/temporális tábla (temporary table):

- Ideiglenes ideig tárolódnak (és csak bizonyos *session*-kben láthatóak), aztán törlődnek.
- DML műveletek végrehajtása – minden esetben.

Ideiglenes táblák

- Úgy viselkednek, mint az alaptáblák, csak mégsem. :P
- Háttértárolón tárolódnak (**tempdb** rendszeradatbázisban).
- Láthatóságukat és élettartamukat tekintve - két típusú ideiglenes tábla:
  - Lokális temporális tábla: pl. #tablanev
  - Globális temporális tábla: pl. ##tablanev
- *Létrehozás:* DDL utasítással vagy SELECT utasítással
- *Módosítás, törlés (séma és adatok esetén):* ugyanúgy, mint alaptábláknál.

Ideiglenes táblák - példa

```
Szallitok (SzallKod, Nev, Cim)
Szallit(SzallKod, AruKod, Ar)
```

- Ideiglenes tábla létrehozása:

```
CREATE TABLE #AtlagArak(
    SzallNev varchar(20),
    Atlag int)
GO
```

- Sorok beszúrása az ideiglenes táblába:

```
INSERT INTO #AtlagArak
 SELECT Nev, AVG(Ar)
 FROM Szallit sz JOIN Szallitok s
  ON sz.SzallKod=s.SzallKod
 GROUP BY s.SzallKod, Nev
```

Nem csak temporális táblák esetén működik.

Ideiglenes táblák - példa

- Nem szükséges előre létrehozni az ideiglenes táblát.

```
SELECT Nev, AVG(Ar) [AS] Atlag INTO #AtlagArak
FROM Szallit sz JOIN Szallitok s
   ON sz.SzallKod=s.SzallKod
GROUP BY s.SzallKod, Nev
```

- `#AtlagArak` sémája = lekérdezés eredményrelációjának sémája!

```
Szallitok (SzallKod, Nev, Cim)
Szallit(SzallKod, AruKod, Ar)
```

Ideiglenes táblák létrehozása és adatokkal való feltöltése

- Példa:
  - 1.lehetőség:

- Lokális ideiglenes tábla létrehozása:

```
CREATE TABLE #ErtekesitokMasolat(
  EID INT,
  CNP CHAR(13),
  Vezeteknev VARCHAR(100),
  Keresztnev VARCHAR(100),
  KereskedesID INT,
  Fizetes REAL,
  Jutalek REAL)
```

- Sorok beszúrása az ideiglenes táblába:

```
INSERT INTO #ErtekesitokMasolat
  SELECT *
  FROM Autoertekesitok
```

Nem csak temporális táblák esetén működik.

Autoertekesitok(EID, CNP, Vezeteknev, Keresztnev, *KereskedesID*, Fizetes, Jutalek)

Ideiglenes táblák létrehozása és adatokkal való feltöltése

- Példa:
- 2.lehetőség:
  - Ideiglenes tábla létrehozása és feltöltése egyetlen utasítással:

```
SELECT * INTO #ErtekesitokMasolat
FROM Autoertekesitok
```

  - Ez esetben: ideiglenes tábla sémája = lekérdezés eredményrelációjának sémája!

Autoertekesitok(EID, CNP, Vezeteknev, Keresztnev, *KereskedesID*, Fizetes, Jutalek)

Ideiglenes táblák

- Lokális temporális táblák (#) :
  - hatásköre: limitált a létrehozó session-re (pl. tárolt eljárás vagy egymásba ágyazott tárolt eljárások).
  - törlésük: automatikus – session- vagy a tárolt eljárás végrehajtásának befejezésékor.
- Globális temporális táblák (##)
  - törlésük: automatikus – létrehozó session bezárásakor

- Előnyük *(mindkettőnél)* alaptábla használatával szemben:
  - a zárolások számának csökkenése

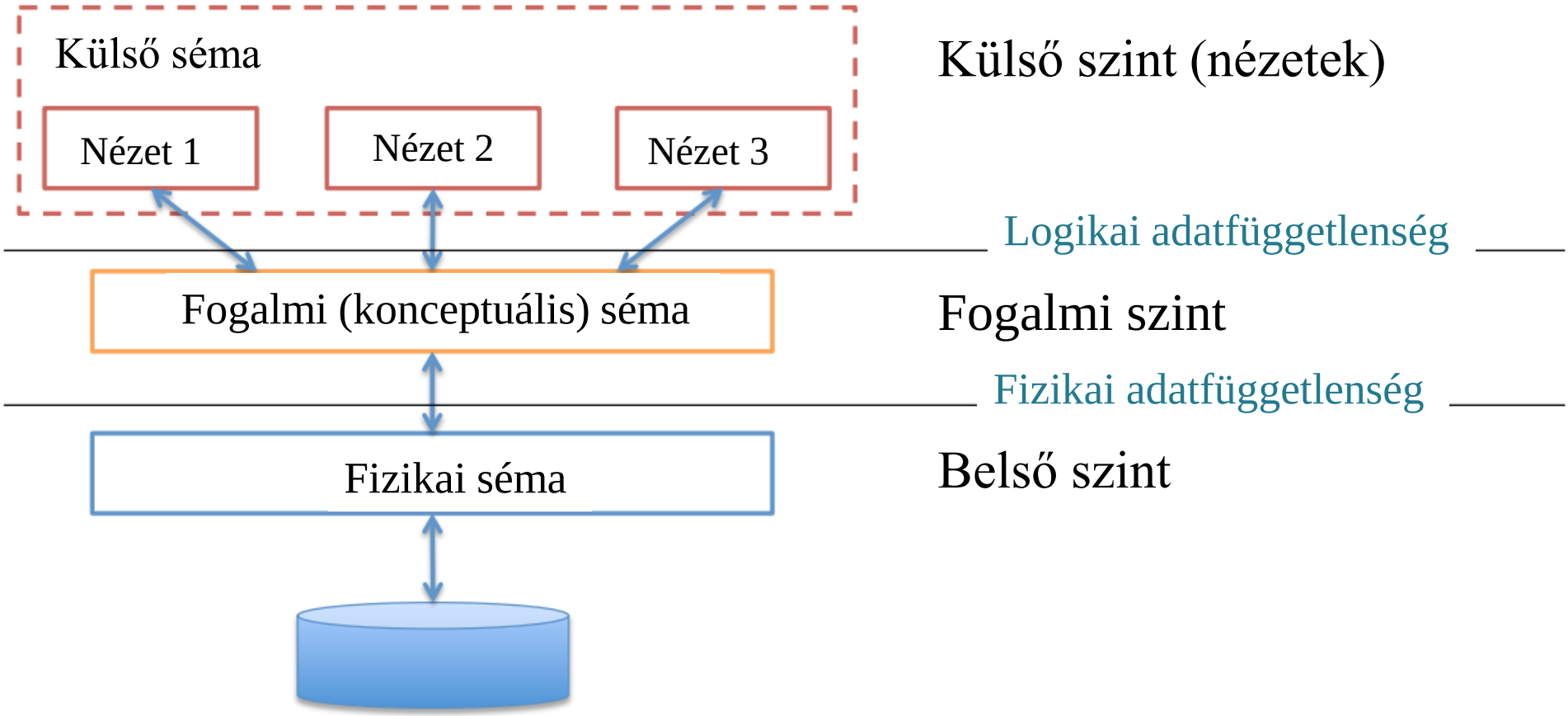
    (aktuális felhasználó ↔ elérheti a táblát-*ő egyedül*)
  - kevesebb log-olás.

Relációtípusok SQL-ben

3. Nézet(tábla) (view):

- Számításokból kapott relációk.
- Az eredmény nem tárolódik az adatbázisban, csak a nézet értelmezése.
- Egyes esetekben módosíthatjuk is őket.

Kitérő:
Adatbázisok ANSI/SPARC architektúrája



Nézettáblák (Views)

- Nézet (view) ⟷ egy ablaka az adatoknak:
  - Ha az adatok változnak, mely ezen részre vonatkozik: a változás látszik a nézetben (ablakban) is.
  - Ha az ablakban változtatunk, a változtatást elvégzi a megfelelő táblában, ha a nézet módosítható.
- A nézet eredménye: reláció ↔ származtatott (derived) reláció.
- A rendszer csak a nézet értelmezését tárolja, az eredményét nem.
  - Ahányszor szüksége van rá, végrehajtja a SELECT parancsot a nézet értelmezéséből.
  - A nézet eredményét nem tárolja. ⟹ A nézet egy „virtuális reláció”.

Nézettáblák (Views)

- Nézet létrehozásának általános szintaxisa:

```
CREATE VIEW <nézet_név>[(<attribútumnevek>)]
 AS <SELECT _SQL_parancs>
 [WITH CHECK OPTION [CONSTRAINT megszorítás]]
 [WITH READ ONLY]
```

- A `CREATE VIEW` parancsba egy alkérdést ágyazunk.
- Az alkérdés tartalmazhat komplex `SELECT` parancsot.
- Az alkérdés NEM tartalmazhat `ORDER BY` záradékot.

Nézettábla létrehozása

- Példa: *Értelmezzünk egy nézetet, mely a kolozsvári kereskedésekben dolgozó értékesítők információit tartalmazza, a keresetüket is tárolva egy származtatott attribútumban!*

```
CREATE VIEW vKolozsvariErtekesitok AS
  SELECT CNP, Vezeteknev, Keresztnev,
         Fizetes+Jutalek AS Kereset
  FROM Autoertekesitok a JOIN Kereskedesek k
      ON a.KereskedesID = k.KereskedesID
  WHERE Varos = 'Kolozsvár'
```

Kereskedesek (KereskedesID, Knev, Varos)
Autoertekesitok(EID, CNP, Vezeteknev, Keresztnev, *KereskedesID*, Fizetes, Jutalek)

Nézettábla lekérdezése

- Példa: Azon kolozsvári autóértékesítők adatait jelenítsük meg, akiknek a keresete nagyobb, mint 8000RON!

```
SELECT *
FROM vKolozsvariErtekesitok
WHERE Kereset > 8000
```

- A lekérdezés végrehajtása esetén a rendszer a nézettábla nevét helyettesíti az értelmezésével.

vKolozsvariErtekesitok(EID, Vezeteknev, Keresztnev, Kereset)

Nézettábla értelmezésének módosítása és törlése

- Nézettábla értelmezésének módosítása: `ALTER VIEW` segítségével.
- Példa: `vKolozsvariErtekesitok` nézet értelmezése esetén néhány oszlopnak új név megadása.

```
CREATE VIEW vKolozsvariErtekesitok(ESzemSzam, ENev,
                                   EKereset) AS
  SELECT ECNP, CONCAT(Vezeteknev,' ',Keresztnev),
         Fizetes+Jutalek
  FROM Autoertekesitok a JOIN Kereskedesek k
           ON a.KereskedesID = k.KereskedesID
  WHERE Varos = 'Kolozsvár'
```

- `CREATE/ALTER VIEW`-ban: oszlopnevek sorrendje ↔ alkérdésbeli oszlopnevek sorrendje.

Nézettábla értelmezésének módosítása és törlése

- Nézettábla értelmezésének törlése: `DROP VIEW` segítségével.

Példa: `DROP VIEW vKolozsvariErtekesitok`

Ellenőrzéssel:

```
IF OBJECT_ID('<nézet_név>', 'V') IS NOT NULL
    DROP VIEW <nézet_név>
```

VAGY:

```
IF EXISTS (
    SELECT * FROM sys.objects
    WHERE object_id = OBJECT_ID(N'<nézet_név>')
                AND type = (N'V'))
 DROP VIEW <nézet_név>
```

- Törölve a nézet értelmezését: az adatok nem veszlődnek el, mivel azok az alaptáblákban vannak, a nézet csak virtuális reláció.

Nézettábla értelmezésének módosítása és törlése

SQL 2016-tól: `IF EXISTS` argumentum használatával:

```
DROP VIEW [IF EXISTS]
    {[schema_name.]<nézet_név>} [ ,...n ]
```

Példa: `DROP VIEW IF EXISTS vKolozsvariErtekesitok`

Mire használhatjuk a nézeteket?

- Hozzáférés korlátozása az adatbázishoz
  - Jogok kiadása a felhasználóknak – nézetekre és nem az alaptáblákra.
- Komplex lekérdezések egyszerűbbé tétele (makro lehetőség). (pl. *V1)* feladat)
- Adatfüggetlenség biztosítása
  - A felhasználó nem érzékeli, ha a fogalmi adatbázis szerkezete változik, vagyis ha bővül (új attribútumok) vagy átszerveződik (új táblák).
- Lehetőség, hogy ugyanannak az adatnak különböző nézeteit mutassuk a különböző felhasználóknak.

Adatkezelési műveletek végrehajtása egy nézeten

- Egyszerű nézeteken – adatkezelési műveletek (DML) végrehajthatóak.
- Nem törölhetünk sorokat egy nézetből, ha tartalmaz:
  - összesítő függvényt;
  - `GROUP BY` záradékot;
  - `DISTINCT` kulcsszót.
- Nem módosíthatjuk a nézet adatait, ha a nézet tartalmaz:
  - a törlésnél felsorolt feltételeket;
  - oszlopokat, melyek kifejezésként vannak értelmezve.

Adatkezelési műveletek végrehajtása egy nézeten

- Nem tudunk adatokat beszúrni, ha:
  - A nézet tartalmazza valamely eddigi feltételt.
  - Vannak olyan `NOT NULL` oszlopok az alaptáblákban (`DEFAULT` nélkül), melyek nincsenek kiválasztva a nézetben (pl. PK).

Feladatok

Országok(OKod, ONev)
Helysegek(HKod, HNev, *OKod*)
Szállástípusok(SzTID, SzTNev)
Szállások(SzID,SzNev,*Hkod*,*SztID*,*Csillag*,*NapiAr*)
Turisták(TID, TNev, Email, *HKod*)
Foglalasok(*SzID*,*TID*,KezdDatum,NapokSzama,Osszar)

Oldjuk meg a feladatokat nézetek és temporális táblák használatával is! Ahol érdemes, az értékeket mentsük el változókban!

V1) Adjuk meg azon szálláshelyeket, ahol történt legalább két foglalás az elmúlt hónapban vagy Kolozsváron találhatóak!

V2) Adjuk meg azokat a helységeket, ahol egyetlen budapesti turista sem járt (⇔ soha nem járt budapesti turista)!

Feladatok

Országok(OKod, ONev)
Helysegek(HKod, HNev, *OKod*)
Szállástípusok(SzTID, SzTNev)
Szállások(SzID,SzNev,*Hkod,SztID,Csillag,NapiAr*)
Turisták(TID, TNev, Email, *HKod*)
Foglalasok(*SzID,TID*,KezdDatum,NapokSzama,Osszar)

Oldjuk meg a feladatokat nézetek és temporális táblák használatával is! Ahol érdemes, az értékeket mentsük el változókban!

V3) Keressük azon helységeket, ahol minden szállástípusból található szállás!

V4) Keressük azon helységeket, ahol legalább azon szállástípusok megtalálhatóak, mint Kolozsváron!