SQL

Molnár Andrea BBTE kurzus

Adatbázisok (amelyeket használni fogunk)

A. Tekintsük a *Kereskdelem* nevű adatbázist a köv. rel.sémákkal:

Részlegek(RészlegID, Név, Helység,
ManSzemSzám);

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

Managerek(*SzemSzám*);

ÁruCsoportok(CsopID, Név, *RészlegID*);

Áruk(ÁruID,Név,MértEgys,MennyRakt,*CsopID*);

Szállítók(SzállID, Név, Helység, UtcaSzám);

Adatbázisok (amelyeket használni fogunk)

A.

Kereskedelmi adatbázis (folyt.):

Vevők(VevőID,Név,Helység,UtcaSzám,Mérleg,
Hihetőség);

Szállít(*SzállID, ÁruID*, Ár);

Szerződések(SzerződID, Dátum, Részletek,
VevőID);

Tételek(TételID, Dátum, *SzerződID*);

Szerepel(*TételID, ÁruID*, RendMenny,
SzállMenny)

Adatbázisok (amelyeket használni fogunk)

B.

Tekintsük az *Egyetem* nevű adatbázist a köv. rel.sémákkal:

```
Szakok(SzakKód, SzakNév, Nyelv);
Csoportok(CsopKód, Evfolyam, SzakKód);
Diákok(BeiktatásiSzám, Név, SzemSzám, Cím,
          SzületésiDatum, CsopKód, Átlag);
TanszékCsoportok(TanszékCsopKód, Név);
Tanszékek(TanszékKód, Név, TanszékCsopKód);
```

Adatbázisok (amelyeket használni fogunk)

B.

Egyetem adatbázis (folyt.): Beosztások(BeosztásKód, Név);

Tanárok(TanárKód, Név, SzemSzám, Cím, PhD, *TanszékKód, BeosztásKód*,Fizetés);

Tantárgyak(TantKod, Név); Tanít(*TanárKod, TantKod*);

Jegyek(JegyID, *BeiktatásiSzám, TantKód*, Datum, Jegy)

Adatok lekérdezése ([D]QL)
Egyszerű lekérdezések SQL-ben

- Legyen $R(A_1, A_2, ..., A_n)$ reláció (tábla). Az alábbi lekérdezés:

  SELECT $A_{i_1}, A_{i_2}, \ldots, A_{i_k}$

  FROM R

  megfelel a relációs algebra vetítés műveletének:

  $$\pi_{A_{i_1}, A_{i_2}, \ldots, A_{i_k}}(R)$$

- A SELECT kulcsszó után megadhatjuk az R relációnak a lekérdezés eredményében megjeleníteni kívánt attribútumait.
  - Az eredmény sorok csak ezen attribútumokat fogják tartalmazni, ***ugyanazzal a névvel***, *amivel az R relációban szerepelnek.*
- A FROM kulcsszó után adhatjuk meg az(oka)t a reláció(ka)t, mely(ek)re a lekérdezés vonatkozik.

Egyszerű lekérdezések SQL-ben

- A relációs algebra kiválasztás művelete:

  $\sigma_p(R)$, ahol ***R*** - reláció, ***p*** - szelekciós feltétel

- A kiválasztás feltételét a WHERE kulcsszó után tudjuk megadni.

```
SELECT *
FROM R
WHERE p
```

- A SELECT kulcsszó utáni * jelentése: „összes attribútum” (a FROM után megadott reláció összes attribútuma)

Példák

- *Adjuk meg az alkalmazottak nevét és fizetését!*

$\pi_{\text{Név,Fizetés}}(\text{Alkalmazottak})$

```
SELECT Név, Fizetés
FROM Alkalmazottak
```

- Mi a különbség a fenti lekérdezés és az alábbiak között? Hogyan írnánk le őket relációs algebrai művelet(ek) segítségével?

```
SELECT *
FROM Alkalmazottak
```

```
SELECT *
FROM Alkalmazottak
WHERE RészlegID=9
   AND Fizetés>500
```

- Az SQL nyelv nem különbözteti meg a kis és nagy betűket (case insensitive) + nem szükséges új sorba írni a SELECT, FROM és WHERE kulcsszavakat. Általános konvenció: a kulcsszavakat nagybetűkkel írjuk + nem használunk ékezeteket.

Egyszerű lekérdezés feldogozása

a) A FROM kulcsszó után megadott relációt a feldolgozó végigjárja.
b) Minden sor esetén ellenőrzi a WHERE kulcsszó után megadott feltételt.
c) Azon sorokat, melyek esetén a feltétel teljesül, az eredmény relációba helyezzük.

- Megj. A feldogozás hatékonyságát növeli, ha a feltételben szereplő attribútumok szerint létezik indexállomány.

Összehasonlító operátorok

- A keresési feltételben (WHERE) szerepelhetnek összehasonlító operátorok, melyek numerikus, karakteres és dátum típusú adatok esetén is használhatóak.
- Az összehasonlító operátorok segítségével attribútumokat és konstansokat hasonlíthatunk össze.
- Szöveg és dátum konstansok megadása: idézőjelek között.

Összehasonlító operátorok

- *Karakterláncok összehasonlítása* esetén használhatjuk a LIKE kulcsszót egy mintával való összehasonlításhoz:

  `k LIKE m`, ahol `k` egy karakterlánc és `m` egy minta.

- A LIKE kulcsszó segítségével képzett feltétel igaz, ha a `k` karakterlánc megfelel az `m` mintának.
- Két speciális karakter a minta esetén:
  - % jel ↔ 0 vagy több tetszőleges karakter
  - _ jel ↔ egyetlen tetszőleges karakter
- Példa:

```
SELECT *
FROM Alkalmazottak
WHERE Név LIKE 'Z_%a';
```

- Használhatjuk a `k NOT LIKE m` kifejezést is.

Sztringműveletek

- AZ SQL támogatja a stringműveleteket is:
  - Összefűzés: `s1+s2` vagy `CONCAT(s1,s2,...,sn)`
  - kisbetű nagybetűvé alakítása (és fordítva):
    `UPPER(s1),LOWER(s1)`
  - sztring hosszának megállapítása: `LEN(s1)`
  - rész-sztring kinyerése:
    `SUBSTRING(sztring,kezdet,hossz)`
  - Szóköz eltüntetése: `LTRIM(sztring), RTRIM(sztring)`
    -A sztring elejéről/végéről levágja a szóközöket.
  - `LEFT, RIGHT, CHARINDEX, LTRIM, RTRIM` stb.
- Példa: *Mit for kiírni?*

```
SELECT LEFT('HELLO WORLD',
            CHARINDEX('W','NEW WORLD', 2))
```

Dátumfüggvények

- Gyakran használt dátumfüggvények SQL-ben:
  - GETDATE() – aktuális dátum
  - YEAR(date), MONTH (date), DAY(date)
  - DATEDIFF(datepart, start, end), datepart lehet pl.: d,m,y
- További dátumfüggvények:
  - EOMONTH – megadott dátumbeli hónap utolsó napja

Összetett feltétel

- A WHERE kulcsszó utáni feltétel lehet összetett:
  - Feltételek összekötése: AND, OR és NOT logikai műveletekkel.
  - SQL-beni erősorrend/megelőzési sorrend: NOT ≫AND ≫OR
  - Ha az erősorrend nem felel meg ⟹A műveletek sorrendjének a meghatározására használhatunk zárójeleket.
- Példa: *„Keressük a 3-as és 6-os részleg alkalmazottait, akiknek fizetése kisebb, mint 200 €.”*

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

Összetett feltétel

- A WHERE kulcsszó utáni feltétel lehet összetett:
  - Feltételek összekötése: AND, OR és NOT logikai műveletekkel.
  - SQL-beni erősorrend/megelőzési sorrend: NOT ≫AND ≫OR
  - Ha az erősorrend nem felel meg ⟹A műveletek sorrendjének a meghatározására használhatunk zárójeleket.
- Példa: *„Keressük a 3-as és 6-os részleg alkalmazottait, akiknek fizetése kisebb, mint 200 €.”*

```
SELECT Név, Fizetés
FROM Alkalmazottak
WHERE (RészlegID = 3 OR RészlegID = 6)
      AND Fizetés < 200;
```

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

NULL értékek

- SQL rendszerek „háromértékű logikát” használnak:
  - TRUE, FALSE + **UNKNOWN** (definiálatlan érték)

AND	*False*	*Null*	*True*
False	F (0)	F (0)	F (0)
Null	F (0)	N (½)	N (½)
True	F (0)	N (½)	T (1)

OR	*False*	*Null*	*True*
False	F (0)	N (½)	T (1)
Null	N (½)	N (½)	T (1)
True	T (1)	T (1)	T (1)

NOT	*False*	*Null*	*True*
	T (1)	N (½)	F (0)

Praktikusan: **False := 0 ; Null := 0.5 ; True := 1**

Ekkor: *P* **AND** *Q* == **MIN**(*p*, *q*) ; *P* **OR** *Q* == **MAX**(*p*, *q*) ; **NOT**(*P*) == **1** – *p* .

- SQL-szabvány szerint: egy logikai kifejezés értéke ISMERETLEN (UNKNOWN), ha benne NULL érték szerepel.
  - NULL * 0; RészlegID = NULL → logikai értékük: NULL
- Egy WHERE-beli állítás értékét hamisnak tekintjük akkor is, ha a kifejezés értéke „ismeretlen”.

NULL értékek

- Vizsgálata: x IS (NOT) NULL: igaz, ha az x mező értéke (nem) NULL
  - Helytelen: x=NULL (NULL nem egy konstans)

pl.
```
SELECT ReszlegNev
FROM Reszlegek
WHERE ManSzemSzam IS NULL
```

NULL értékek

- Vizsgálata: x IS (NOT) NULL: igaz, ha az x mező értéke (nem) NULL
  - Helytelen: x=NULL (NULL nem egy konstans)

  pl.
```
SELECT ReszlegNev
FROM Reszlegek
WHERE ManSzemSzam IS NULL
```

  - Azon részlegek nevét adja meg, amelyeknek nincs managerük.
- Háromértékű logikában nem teljesül: A v ¬A≠ I

NULL értékek

- Vizsgálata: x IS (NOT) NULL: igaz, ha az x mező értéke (nem) NULL
  - Helytelen: x=NULL (NULL nem egy konstans)

pl.
```
SELECT ReszlegNev
FROM Reszlegek
WHERE ManSzemSzam IS NULL
```

  - Azon részlegek nevét adja meg, amelyeknek nincs managerük.
- Háromértékű logikában nem teljesül: A v ¬A≠ I

pl.
```
SELECT SzallID, AruID
FROM Szallit
WHERE Ar > 50 OR Ar <= 50
```

  - Nem jelenik meg az eredményben az az áru, amelyikhez nincs szállítás rendelve.

Aritmetikai kifejezések

- A SELECT záradék tartalmazhat *aritmetikai kifejezéseket* is (használható operátorok: +, –, ×és /), konstansokra vagy sorok attribútumaira vonatkozóan.
- Pl.
```
SELECT Név, Fizetés*1.3
FROM Alkalmazottak
    WHERE (RészlegID = 2 OR RészlegID = 9)
            AND Fizetés > = 500
```
  - Az eredmény relációban *a 2-es és 9-es részleg nagyfizetésű alkalmazottainak neve jelenik meg, valamint fizetésük $-ban (felt.: €/$ = 1.3).*

Aritmetikai kifejezések

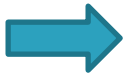
- Pl.
```
SELECT Név, Fizetés*1.3
FROM Alkalmazottak
    WHERE (RészlegID = 2 OR RészlegID = 9)
                AND Fizetés > = 500
```

Szemszám	*Név*	*RészlegID*	*Fizetés*
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
567765	Katona József	NULL	600
556789	Kovács Anna	NULL	700
333333	Kovács István	2	500

Aritmetikai kifejezések

- Pl.
```
SELECT Név, Fizetés*1.3
FROM Alkalmazottak
    WHERE (RészlegID = 2 OR RészlegID = 9)
            AND Fizetés > = 500
```

Szemszám	*Név*	*RészlegID*	*Fizetés*
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
567765	Katona József	NULL	600
556789	Kovács Anna	NULL	700
333333	Kovács István	2	500



A rendszer nem ad nevet a kiszámított attribútumnak.

Név	*(No column name)*
Szabó János	1170
Szilágyi Pál	910
Kovács István	650

Átnevezés

- ***AS** kulcsszó*: a vetítés során kapott eredmény relációban az *attribútumok nevének megváltoztatására*, ha a FROM után szereplő reláció attribútumainak nevei nem felelnek meg; VAGY ha az adott oszlop értékei valamilyen művelet eredményei lesznek (pl. összesítő függvények vagy aritmetikai operátorok esetén). Használata: nem kötelező. Másik lehetőség: "=".
- Előbbi példa átnevezéssel:

```
SELECT Név AS Nev9,
   Fizetes$=Fizetés*1.3
FROM Alkalmazottak
WHERE (RészlegID = 2 OR RészlegID = 9)
                AND Fizetés > = 500
```

Név9	Fizetes$
Szabó János	1170
Szilágyi Pál	910
Kovács István	650

Más szűrőfeltételek

- BETWEEN kulcsszó: segítségével megadunk egy intervallumot, és azt vizsgáljuk, hogy adott oszlop mely értéke esik a megadott intervallumba. Általános alakja:

```
WHERE <oszlop> BETWEEN <kifejezés_1> AND
                       <kifejezés_2>
```

- **Példa:**

```
SELECT Név
FROM Alkalmazottak
WHERE Fizetés BETWEEN 300 AND 500;
```

- Ugyanazt az eredményt adja, mint a:

```
SELECT Név
FROM Alkalmazottak
WHERE Fizetés >= 300 AND Fizetés <= 500;
```

Más szűrőfeltételek

- Az IN operátor után megadunk egy értéklistát, és azt vizsgáljuk, hogy az adott oszlop mely mezőinek értéke egyezik az adott lista valamelyik elemével. Általános alakja:

```
WHERE <oszlop> IN <kifejezés_1>,<kifejezés_2>[,...])
```

- **Példa:** Tekintsük a következő relációt:

```
Diákok (BeiktatásiSzám, Név, Cím,
        SzületésiDatum, CsopKod, Átlag);
```

„Keressük az ’531’-es, ’532’-s és ’631’-es csoportok diákjait:”

```
SELECT Név
FROM Diákok
WHERE CsopKod IN ('531', '532', '631');
```

Eredmény reláció sorainak rendezése

- ORDER BY kulcsszóval
- Növekvő sorrendbe rendezés (alapértelmezés szerinti) - ASC kulcsszóval, csökkenő sorrend: DESC kulcsszóval.
- Előbbi példa rendezéssel:

```
SELECT Név
FROM Diákok
WHERE CsopKod IN ('531', '532', '631')
ORDER BY CsopKod, Név;
```

- **Példa:**

```
SELECT Név, Átlag
FROM Diákok
ORDER BY 2 DESC;
```

Eredmény reláció sorainak rendezése

- ORDER BY kulcsszóval
- Növekvő sorrendbe rendezés (alapértelmezés szerinti) - ASC kulcsszóval, csökkenő sorrend: DESC kulcsszóval.
- Előbbi példa rendezéssel:

```
SELECT Név
FROM Diákok
WHERE CsopKod IN ('531', '532', '631')
ORDER BY CsopKod, Név;
```

- **Példa:** A diákokat átlag szerint csökkenő sorrendben adja meg!

```
SELECT Név, Átlag
FROM Diákok
ORDER BY 2 DESC;
```

ORDER BY n [ASC/DESC]

n - a vetítésben szereplő n. attribútum

Több relációra vonatkozó lekérdezések

- Relációs algebra *fő tulajdonsága*: a műveletek eredménye **reláció**, és az **eredmény operandus lehet a következő műveletben**.
- A SELECT-SQL kihasználja ezt: a relációkat össze-kapcsolhatjuk, egyesíthetjük, metszetet vagy különbséget is számíthatunk.
- Relációk összekapcsolásakor meg kell adni az összekapcsolás módját (belső vagy külső) és a sorok összekapcsolásának feltételét.
- Általánosan:

```
SELECT <attribútumok_lista>
FROM <relációk_lista>
WHERE <feltételek>
```

Több relációra vonatkozó lekérdezések

- A Descartes-szorzat művelete:

  $\boldsymbol{R \times S}$

- Megvalósítás SQL parancs segítségével:

```
SELECT *
FROM R, S;
```

- Theta-összekapcsolás: $R \bowtie_{\theta} S$

  Megvalósítás SQL-ben:

```
SELECT <attribútumok>
FROM R, S
 WHERE θ;
```

- Természetes összekapcsolás:

$$R \bowtie S = \pi_{B \cup C}\left(R \bowtie_{(R.A_1 = S.A_1 \wedge R.A_2 = S.A_2 \wedge \ldots \wedge R.A_p = S.A_p)} S\right)$$

  SQL-ben:

```
SELECT <attribútumok>
FROM R, S
WHERE R.A1=S.A1 AND ... AND R.Ap=S.Ap;
```

  - Leggyakrabban használt művelet.

Több relációra vonatkozó lekérdezések

- Példa: Tekintsük (ismét) a következő relációkat:

  Csoportok(CsopKod, Evfolyam, *SzakKod*);

  Diákok(BeiktatásiSzám, Név, Cím,

  SzületésiDatum, *CsopKod*, Átlag);

- *Irassuk ki a diákok esetén az évfolyamot és szakkódot is!*

```
SELECT Nev, CsopKod, Evfolyam, SzakKod
FROM Diakok, Csoportok
   WHERE Diakok.CsopKod = Csoportok.CsopKod;
```

- **Mi történik, ha elfelejtjük a join feltételt?**

Több relációra vonatkozó lekérdezések

- Példa: Tekintsük (ismét) a következő relációkat:

  Csoportok(CsopKod, Evfolyam, *SzakKod*);

  Diákok(BeiktatásiSzám, Név, Cím,

  SzületésiDatum, *CsopKod*, Átlag);

- *Irassuk ki a diákok esetén az évfolyamot és szakkódot is!*

```
SELECT Nev, CsopKod, Evfolyam, SzakKod
FROM Diakok, Csoportok
   WHERE Diakok.CsopKod = Csoportok.CsopKod;
```

- **Mi történik, ha elfelejtjük a join feltételt? ⟹ Az eredmény Descartes-szorzat lesz**, melynek méretei nagyon nagyok lehetnek.

Több relációra vonatkozó lekérdezések

- Az előbb feladat megoldása (INNER) JOIN kulcsszó megadásával:

```
SELECT Nev, CsopKod, Evfolyam, SzakKod
FROM Diakok
[INNER]JOIN Csoportok
  ON Diakok.CsopKod = Csoportok.CsopKod;
```

- Külső (outer) join is meg van valósítva SQL-ben *(részletek mindjárt)*

Kiválasztás műveletének megadása több reláció összekapcsolása mellett

`Csoportok(CsopKod, Evfolyam, SzakKod);`

`Diakok(BeiktatasiSzam, Nev,...,CsopKod,...)`

- *Keressük a harmadéves diákok nevét!*

Kiválasztás műveletének megadása több relá összekapcsolása mellett

Csoportok(CsopKod, Evfolyam, *SzakKod*);

Diakok(BeiktatasiSzam, Nev,...,*CsopKod*,...)

- *Keressük a harmadéves diákok nevét!*

```
SELECT Nev
FROM Diakok, Csoportok
WHERE Diakok.CsopKod = Csoportok.CsopKod
    AND Evfolyam = 3;
```

Másképp (JOIN kulcsszó használatával):

Kiválasztás műveletének megadása több reláció összekapcsolása mellett

Csoportok(CsopKod, Evfolyam, *SzakKod*);

Diakok(BeiktatasiSzam, Nev,...,*CsopKod*,...)

- *Keressük a harmadéves diákok nevét!*

```
SELECT Nev
FROM Diakok, Csoportok
WHERE Diakok.CsopKod = Csoportok.CsopKod
    AND Evfolyam = 3;
```

Másképp (JOIN kulcsszó használatával):

```
SELECT Nev
FROM Diakok JOIN Csoportok
  ON Diakok.CsopKod = Csoportok.CsopKod
WHERE Evfolyam = 3;
```

Kettőnél több reláció összekapcsolása

- Több mint két relációt is összekapcsolhatunk természetes összekapcsolással, fontos, hogy az összes join feltételt megadjuk.
- Ha az összekapcsolandó relációk száma *k* és *minden két-két relációnak egy-egy közös attribútuma* van $\Longrightarrow$join feltételek száma ...

Kettőnél több reláció összekapcsolása

- Több mint két relációt is összekapcsolhatunk természetes összekapcsolással, fontos, hogy az összes join feltételt megadjuk.
- Ha az összekapcsolandó relációk száma *k* és *minden két-két relációnak egy-egy közös attribútuma* van $\Longrightarrow$join feltételek száma *k–1*.

- Tekintsük a következő relációt is:

  `Szakok (SzakKod, SzakNev);`

  Keressük a harmadéves közgazdász diákok nevét!

Kettőnél több reláció összekapcsolása

- *Keressük a harmadéves közgazdász diákok nevét!*

```
SELECT Nev
FROM Diakok, Csoportok, Szakok
WHERE Diakok.CsopKod = Csoportok.CsopKod
    AND Csoportok.SzakKod = Szakok.SzakKod
    AND Evfolyam = 3
    AND SzakNév LIKE '%közgazdász%';
```

Kettőnél több reláció összekapcsolása

- *Keressük a harmadéves közgazdász diákok nevét!*

  JOIN kulcsszó használatával:

```
SELECT Nev
FROM Diakok JOIN Csoportok
  ON Diakok.CsopKod = Csoportok.CsopKod
JOIN SZAKOK
  ON Csoportok.SzakKod = Szakok.SzakKod
WHERE Evfolyam = 3
    AND SzakNév LIKE '%közgazdász%';
```

Feladat

Szállásfoglalásokkal menedzselő cég/weboldal részleges adatbázisa:

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

Relációs algebrai műveleteket tartalmazó kifejezésekkel **+ SELECT utasításokkal** fejezzük ki a következő lekérdezéseket!*

Q1) Adjuk meg a kolozsvári turisták nevét, akik Prágában foglaltak szállást idén augusztusban!

Feladat

Szállásfoglalásokkal menedzselő cég/weboldal részleges adatbázisa:

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

Relációs algebrai műveleteket tartalmazó kifejezésekkel **+ SELECT utasításokkal** fejezzük ki a következő lekérdezéseket!

Q1) Adjuk meg a kolozsvári turisták nevét, akik Prágában foglaltak szállást idén augusztusban! ← több reláció összekapcsolása

Több relációra vonatkozó lekérdezések

- Tekintsük a következő relációt:

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

Keressük azon alkalmazottakat, akik ugyanazon a címen laknak (pl. férj és feleség, vagy szülő és gyerek)!

Több relációra vonatkozó lekérdezések

- *Keressük azon alkalmazottakat, akik ugyanazon a címen laknak (pl. férj és feleség, vagy szülő és gyerek)!*

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
  AND A1.Név < A2.Név;
```

- Sorváltozó (`A1`,`A2`): a FROM záradékban szereplő relációhoz hozzárendelt másodnév (másodlagos név).
  - Használata: ha rövidebb vagy más nevet akarunk adni a relációnak, VAGY ha a FROM után kétszer (vagy többször) is szerepel ugyanaz a reláció, VAGY ha korrelált alkérdésben többször is megjelenik ugyanaz a reláció *(lsd. következő kurzuson)*.
  - Ha definiálunk másodnevet, az adott lekérdezésen belül kötelező azt használni.

Több relációra vonatkozó lekérdezések

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
   AND A1.Név < A2.Név;
```

- A lekérdezés feldolgozó ugyanazt a relációt kell kétszer bejárja, hogy a kért párokat megtalálja.
- *Mi történik, ha az A1.Név < A2.Név feltételt nem adjuk meg?*

Több relációra vonatkozó lekérdezések

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
   AND A1.Név < A2.Név;
```

- A lekérdezés feldolgozó ugyanazt a relációt kell kétszer bejárja, hogy a kért párokat megtalálja.
- *Mi történik, ha az A1.Név < A2.Név feltételt nem adjuk meg?* Minden alkalmazott bekerülne az eredménybe önmagával is párosítva.
- *Mi lenne az eredmény A1.Név < A2.Név helyett A1.Név <> A2.Név feltétel megadásakor?*

Több relációra vonatkozó lekérdezések

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
  AND A1.Név < A2.Név;
```

- A lekérdezés feldolgozó ugyanazt a relációt kell kétszer bejárja, hogy a kért párokat megtalálja.
- *Mi történik, ha az A1.Név < A2.Név feltételt nem adjuk meg?* Minden alkalmazott bekerülne az eredménybe önmagával is párosítva.
- *Mi lenne az eredmény A1.Név < A2.Név helyett A1.Név <> A2.Név feltétel megadásakor?* Egy férj–feleség páros kétszer is bekerült volna, csak más sorrendben.
  - Pl: (‘Kovács István’, ‘Kovács Sára’) és (‘Kovács Sára’, ‘Kovács István’) is.

Több relációra vonatkozó lekérdezések

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
  AND A1.Név < A2.Név;
```

- *Milyen esetben okozhat problémát az `A1.Név < A2.Név` feltétel?*

Több relációra vonatkozó lekérdezések

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
  AND A1.Név < A2.Név;
```

- *Milyen esetben okozhat problémát az* `A1.Név < A2.Név` *feltétel?* Pl. ha a gyereknek lehet ugyanaz a neve, mint a szülőnek.

  $\Longrightarrow$Jobb megoldás: `A.Név < A.Név` feltétel kicserélése `A1.SzemSzám < A2.SzemSzám`-ra.

Több relációra vonatkozó lekérdezések

```
SELECT A1.Név AS Név1, A2.Név AS Név2
FROM Alkalmazottak AS A1, Alkalmazottak AS A2
WHERE A1.Cím = A2.Cím
  AND A1.Név < A2.Név;
```

- *Milyen esetben okozhat problémát az* `A1.Név < A2.Név` *feltétel?* Pl. ha a gyereknek lehet ugyanaz a neve, mint a szülőnek.

⟹Jobb megoldás: `A.Név < A.Név` feltétel kicserélése `A1.SzemSzám < A2.SzemSzám`-ra.

Egyszerű lekérdezés kiértékelésének algoritmusa

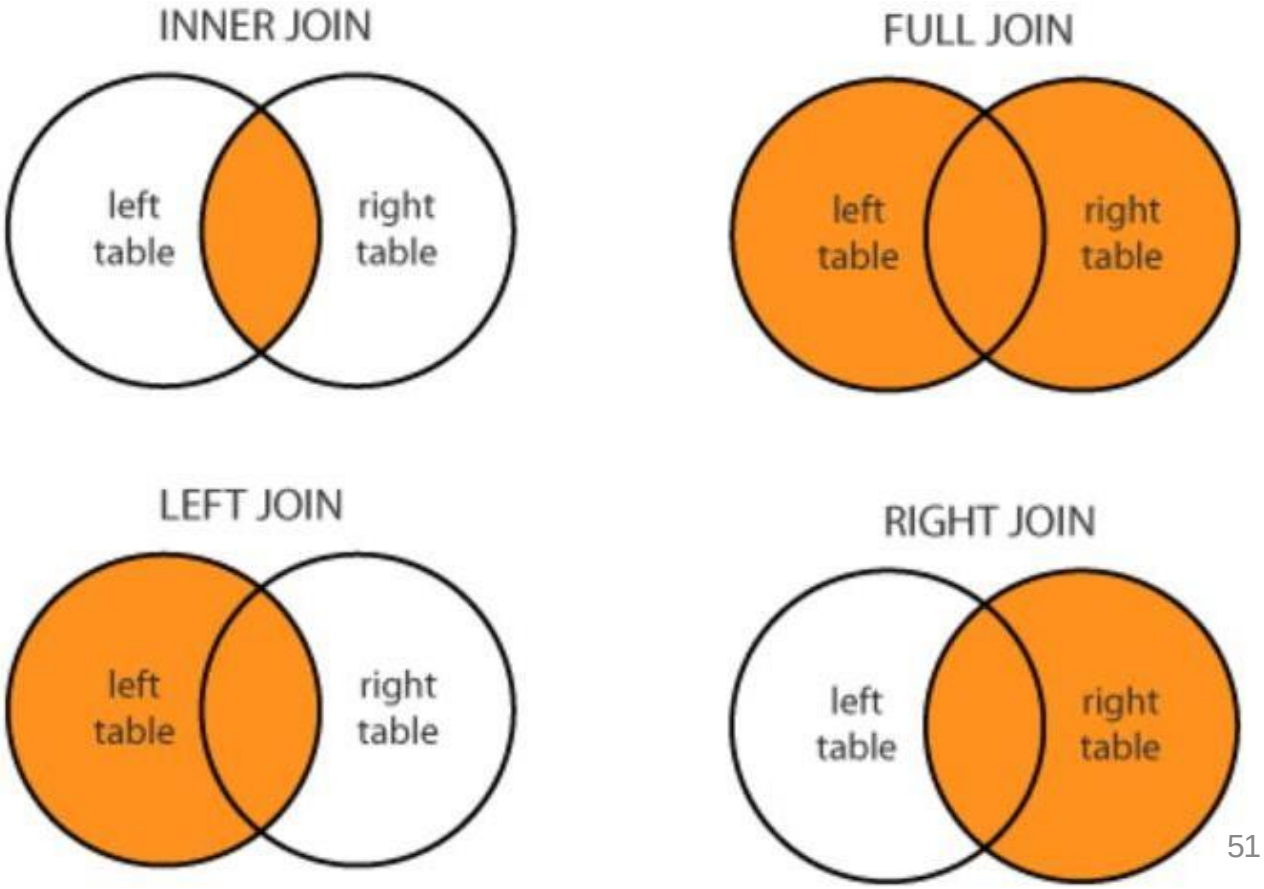
Input: R_1, R_2,…, R_n relációk a FROM záradék után
Begin
  Minden t_1 sorra az R_1-ből
    Minden t_2 sorra az R_2-ből
      …
      Minden t_n sorra az R_n-ből
        Ha a WHERE záradék igaz a t_1, t_2, …, t_n attribútumainak az értékeire
        Akkor
          A SELECT záradék attribútumainak értékeiből alkotott sort az eredményhez adjuk
End

Külső összekapcsolás (OUTER JOIN)

- Join művelet kiterjesztése, mely elkerüli az információvesztést.

⟺Az összekapcsolt táblák egyikénél vagy mindkettőnél garantálja valamennyi rekord megőrzését.

- NULL-okat használ



Külső összekapcsolás (OUTER JOIN)

- Bal oldali külső összekapcsolás:

```
R LEFT [OUTER] JOIN S ON R.X = S.X
```

  - R-nek azon sorai is bekerülnek az eredmény relációba, melyekhez nem kapcsolódik S-beli sor (ahol az X attribútumhalmaz értéke nem létezik az S reláció értékei között. Azon mezők, amelyek csak S-ben szerepelnek, NULL értéket kapnak.
- Jobb oldali külső összekapcsolás:

```
R RIGHT [OUTER] JOIN S ON R.X = S.X
```

  - Lásd LEFT OUTER JOIN-t, csak R és S szerepe megcserélődik.
- Teljes külső összekapcsolás:

```
R FULL [OUTER] JOIN S ON R.X = S.X
```

  - Az eredmény azon sorokat tartalmazza, melyek esetében a közös attribútum értéke megegyezik mindkét relációban + mind a bal oldali R reláció, mind a jobb oldali S reláció lógó sorait magában foglalja. Az ezáltal üresen maradó mezők ez esetben is NULL értéket kapnak.

Példa

Students

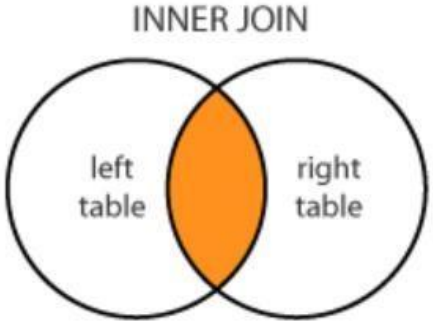
sid	*name*	*email*	*age*	*gr*
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

cid	*cname*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

sid	*cid*	*grade*
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



$\pi_{\text{Students.Name,Courses.Name}}(\text{Students} \bowtie \text{Enrolled} \bowtie \text{Courses})$

```
SELECT S.name, C.cname
FROM Students S
INNER JOIN Enrolled E
ON S.sid = E.sid
INNER JOIN Courses C
ON E.cid = C.cid
```

name	*cname*
John	Algorithms1
Smith	Algorithms1

Példa külső összekapcsolásra

Students

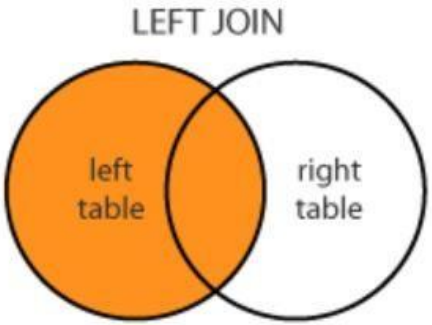
sid	*name*	*email*	*age*	*gr*
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

cid	*cname*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

sid	*cid*	*grade*
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



$\pi_{\text{Students.Name,Courses.Name}}(\text{Students} \supset\!\bowtie \text{Enrolled} \supset\!\bowtie \text{Courses})$

```
SELECT S.name, C.cname
FROM Students S
LEFT OUTER JOIN Enrolled E
                ON S.sid = E.sid,
LEFT OUTER JOIN Courses C
                 ON E.cid = C.cid
```

name	*cname*
John	Algorithms1
Smith	Algorithms1
Anne	**NULL**

Példa külső összekapcsolásra

Students

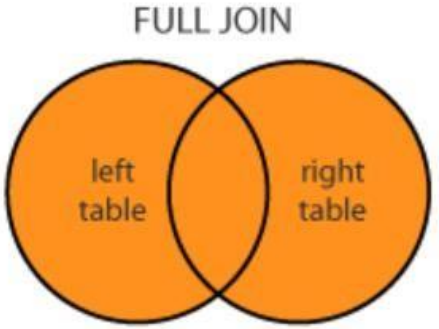
sid	*name*	*email*	*age*	*gr*
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

cid	*cname*	*credits*
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

sid	*cid*	*grade*
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



$\pi_{\text{Students.Name,Courses.Name}}(\text{Students} \overset{\supset\!\subset}{\bowtie} \text{Enrolled} \overset{\supset\!\subset}{\bowtie} \text{Courses})$

```
SELECT S.name, C.cname
FROM Students S
FULL OUTER JOIN Enrolled E
             ON S.sid = E.sid,
FULL OUTER JOIN Courses C
             ON E.cid = C.cid
```

name	*cname*
John	Algorithms1
Smith	Algorithms1
NULL	Databases2
NULL	Databases1
Anne	**NULL**

Feladatok

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

Q2) Adjuk meg a szállások nevét és azok típusát! Azon szállástípus(oka)t is jelenítsük meg, amely(ek)be nem tartozik egy szállás sem!

Q3) Adjuk meg a szállások átlagárát!

Feladatok

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

Q2) Adjuk meg a szállások nevét és azok típusát! Azon szállástípus(oka)t is jelenítsük meg, amely(ek)be nem tartozik egy szállás sem! ← *OUTER JOIN*

Q3) Adjuk meg a szállások átlagárát! ← *összesítés*

Összesítő függvények és csoportosítás

- Általános szintaxis:

```
SELECT [<csoportosító oszlop_lista>],
       [<összesítő függvény>(<oszlop>)]
FROM <reláció_lista>
[WHERE <kifejezés>]
GROUP BY <csoportosító oszlop_lista>
ORDER BY <rendezési attribútum_lista>
```

Összesítő függvények

- Egy oszlopban szereplő értékek összegezésére.
- Az összesítés művelete egy oszlop értékeiből egy új értéket hoz létre.
- Összesítő függvények (*A* - egy oszlopnév):
  - `COUNT(*), COUNT([DISTINCT] A)` – *A* (különböző) értékeinek száma
  - `SUM([DISTINCT] A)` – *A* értékeinek összege
  - `AVG([DISTINCT] A)` – *A* értékeinek átlaga
  - `MAX(A)` – maximális érték *A* értékei közül
  - `MIN(A)` – minimális érték *A* értékei közül
- A reláció egyes sorait bizonyos feltétel szerint csoportosíthatjuk (pl. egy oszlop értéke szerint), és a csoporton belül végezhetünk összesítéseket.

NULL értékek esete az összesítő függvényekkel

- Az összesítő függvények (COUNT(*) kivételével) nem veszik figyelembe azon sorokat, ahol az összesített attribútumok értéke NULL.
- Példa: `SELECT SUM(Fizetés)`
  `FROM Alkalmazottak`
  - Nem veszi figyelembe azon sorokat, ahol Fizetés értéke NULL.
  - Az eredmény NULL, ha egyetlen nem-NULL értékű fizetés sincs.

Feladatok

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

Q3) Adjuk meg a szállások átlagárát!

Q4) Adjuk meg minden helység esetében a turisták számát!

Feladatok

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

Q3) Adjuk meg a szállások átlagárát!

Q4) Adjuk meg minden helység esetében a turisták számát!

← *csoportosítás*

Csoportosítás

- A GROUP BY után megadjuk a csoportosító attribútumok (oszlopok) listáját, melyek azonos értéke szerint történik a csoportosítás.
- **Vigyázat!** Csak ezeket az oszlopokat válogathatjuk ki a SELECT kulcsszó után + azokat, melyekre valamilyen összesítő függvényt alkalmazunk, viszont:
- A SELECT parancs megengedi, hogy a csoportosító attribútum hiányozzon a vetített attribútumok listájából.
- Azon oszlopoknak, melyekre összesítő függvényt alkalmaztunk, érdemes más nevet (lsd. átnevezés) adni, hogy könnyebben tudjunk hivatkozni rá (*a rendszer nem ad nevet a megfelelő oszlopnak az eredmény relációban*).
- Lehetséges több összesítő függvény használata, valamint több csoportosítási attribútum megadása is.

Csoportosítás

- A lekérdezés processzor először rendezi a reláció sorait a csoportosítandó oszlop értékei szerint.
- Utána azokat a sorokat, ahol ezen oszlopoknak ugyanaz az értéke, az eredmény relációban csak egy sor fogja képviselni, ahol megadhatjuk az oszlop értékét, amely a lekérdezett relációban minden sorban ugyanaz.
- A többi oszlopra vonatkozóan csak összesítéseket végezhetünk.

Példák

- *Adjuk meg minden részleg esetén az átlagfizetést, valamint a minimális fizetést!*

```
SELECT ReszlegID, AVG(Fizetes) AtlFiz,
MIN(Fizetes) MinFiz
FROM Alkalmazottak
GROUP BY ReszlegID
```

A kapott eredmény:

RészlegID	*AtlagFiz*	*MinFiz*
1	800	800
2	500	300
9	650	400

Alkalmazottak reláció sorai:

SzemSzám	*Név*	*RészlegID*	*Fizetés*
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
333333	Kovács István	2	500

Példák

Aruk(AruID,Nev,MertEgys,MennyRakt,*CsopID*);
Szallit(*SzallID, AruID*, Ar)

- *Adjuk meg minden áru esetén a szállítók által ajánlott átlagárat!*

```
SELECT Nev, AVG(Ar)
FROM Aruk a, Szallit sz
WHERE a.AruID = sz.AruID
GROUP BY Nev
```

- *Mikor lehet probléma a csoportosítással?*

Példák

`Aruk(AruID,Nev,MertEgys,MennyRakt,CsopID);`

`Szallit(SzallID, AruID, Ar)`

- *Adjuk meg minden áru esetén a szállítók által ajánlott átlagárat!*

```
SELECT Nev, AVG(Ar)
FROM Aruk a, Szallit sz
WHERE a.AruID = sz.AruID
GROUP BY a.AruID, Nev
```

- *Mikor lehet probléma a csoportosítással? – Ha több árunak is ugyanaz a neve.*
  - **Megoldás:** A tábla elsődleges kulcsa szerint csoportosítunk (először).
  - Ha a Nev mező egyedi (UNIQUE), nem szükséges elsődleges kulcs szerint is csoportosítani.

Példák

- Több csoportosítási attribútum megadása is lehetséges.
- Példa:

```
TanszékCsoportok(TanszékCsopKód, Név);
Tanszékek(TanszékKód, Név, TanszékCsopKód);
Beosztások(BeosztásKód, Név);
Tanárok(TanárKód, Név, SzemSzám, Cím, PhD,
                TanszékKód, BeosztásKód,Fizetés);
```

Adjuk meg a tanárok átlagfizetését tanszékeken belül beosztásokra leosztva!

```
SELECT TanszekKod, BeosztasKod, AVG(Fizetes)
FROM Tanarok
GROUP BY TanszekKod, BeosztasKod
```

Példák

Tanár Kod	*Név*	*Cím*	*PhD*	*Beosztás Kod*	*Tanszék Kod*	*Fizetés*
KB12	Kiss Béla	Petőfi u. 12	Y	ADJ	ALG	150
NL03	Nagy László	Kossuth u. 3	Y	ADJ	REN	160
KG05	Kovács Géza	Ady tér 5	N	ADJ	ALG	160
PI14	Péter István	Dóm tér 14	N	TNS	REN	120
NE55	Németh Eva	Dózsa u. 55	Y	PRO	ALG	300
VS77	Vígh Sándor	Rózsa u. 77	Y	PRO	REN	310
LL63	Lukács Lóránt	Viola u. 63	Y	ADJ	REN	170
LS07	László Samu	Rákóczi u. 7	N	TNS	REN	110
KP52	Kerekes Péter	Váczi u. 52	Y	PRO	ALG	280

a lekérdezés eredménye:

Tanszék Kod	*Beosztás Kod*	*AVG (Fizetés)*
ALG	ADJ	155
ALG	PRO	290
REN	ADJ	165
REN	PRO	310
REN	TNS	115

Feladatok

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

Q4) Adjuk meg minden helység esetében a turisták számát!

Q5) Adjuk meg minden szálláshely esetében, hogy a múlt hónapban átlagosan hány napra foglaltak szállást!

Q6) Adjuk meg minden helység esetében, hogy hány különböző szálláshelytípussal rendelkeznek!

COUNT(*) vs COUNT(A) vs. COUNT(DISTINCT A)

Szállások(SzID, SzNev, Hkod, *SztID*, Ar)

Q6)
```
SELECT HKod, COUNT(*) A, COUNT(SztID) B,
       COUNT(DISTINCT SztID) C
FROM Szallasok
GROUP BY HKod
```

SzID	*Hkod*	*SztID*
1	1	1
2	2	NULL
3	1	1
4	2	1
5	2	2
6	3	1

Hkod	*A*	*B*	*C*
1			
2			
3			

COUNT(*) vs COUNT(A) vs. COUNT(DISTINCT A)

Szállások(SzID, SzNev, Hkod, *SztID*, Ar)

Q6)
```
SELECT HKod, COUNT(*) A, COUNT(SztID) B,
       COUNT(DISTINCT SztID) C
FROM Szallasok
GROUP BY HKod
```

SzID	*Hkod*	*SztID*
1	1	1
2	2	NULL
3	1	1
4	2	1
5	2	2
6	3	1

→

Hkod	*A*	*B*	*C*
1	2	2	1
2	3	2	2
3	1	1	1

Mi történik azon helységekkel, amelyekhez nem rendeltünk szálláshelyet?

Ez a (majdnem) helyes.

COUNT(*) vs COUNT(A) vs. COUNT(DISTINCT A)

Szállások(SzID, SzNev, Hkod, *SztID*, Ar)

Q6') OUTER JOIN-nal (ha érdekelnek azok a helységek is, ahol nincs egyetlen szállás(típus) sem):

```
SELECT h.HKod, COUNT(DISTINCT SztID)
FROM Szallasok sz RIGHT JOIN Helysegek h
  ON h.HKod=sz.HKod
GROUP BY h.HKod
```

SzID	Hkod	SztID
1	1	1
2	2	NULL
3	1	1
4	2	1
5	2	2
6	3	1

→

Hkod	A	B	C
1	2	2	1
2	3	2	2
3	1	1	1
4	1	0	0

Feladatok

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

Q7) Adjuk meg azon turisták nevét, akik legalább kétszer foglaltak szállást (valamelyik szálláshelyen)!

Feladatok

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

Q7) Adjuk meg azon turisták nevét, akik legalább kétszer foglaltak szállást (valamelyik szálláshelyen)!

↳ HAVING

Összesítő függvények értékének vizsgálata

- Általános szintaxis:

```
SELECT [<csoportosító oszlop_lista>],
       [<összesítő függvény>(<oszlop>)]
FROM <reláció_lista>
[WHERE <kifejezés>]
GROUP BY <csoportosító oszlop_lista>
ORDER BY <rendezési attribútum_lista>
HAVING <csoportosítási feltétel>
ORDER BY <rendezési attribútum_lista>
```

- Fontos a sorrend!
- HAVING CSAK GROUP BY-al együtt!

Összesítő függvények értékének vizsgálata

- A csoportosítás után kapott eredmény reláció soraira a HAVING kulcsszót használva egy feltételt alkalmazhatunk.
- Ha csoportosítás előtt szeretnénk kiszűrni sorokat, azokra a WHERE feltételt lehet alkalmazni.
- A HAVING kulcsszó utáni feltételben azon oszlopok szerepelhetnek, melyekre a SELECT parancsban összesítő függvényt alkalmaztunk.
- HAVING megkerülhető *(lsd. később)*

Példák

Tanszékek(TanszékKód, Név, *TanszékCsopKód*)

- *Keressük azon tanszékeket, ahol a tanársegédeket kivéve a tanárok átlagfizetése nagyobb, mint 240 euro.*

```
SELECT TanszékKod, AVG(Fizetés)
   FROM Tanárok
   WHERE BeosztásKod <> 'TNS'
   GROUP BY TanszékKod
    HAVING AVG(Fizetés) > 240;
```

Az eredmény a következő lesz:

Tanszék Kod	*AVG (Fizetés)*
ALG	125

- A kiértékelés során a csoportosítás után minden csoportra megnézzük a feltételt és eldobjuk azokat a csoportokat, melyek nem teljesítik azt, s csak a maradékkal dolgozunk tovább.

Példák

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

- *Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500euro, átlagfizetés szerint növekvő sorrendben!*

```
SELECT ReszlegID, AVG(Fizetes)
  FROM Alkalmazottak
  GROUP BY ReszlegID
  HAVING AVG(Fizetes)>500
  ORDER BY AVG(Fizetes)
```

Alkalmazottak reláció sorai:

SzemSzám	*Név*	*RészlegID*	*Fizetés*
111111	Nagy Éva	2	300
222222	Kiss Csaba	9	400
456777	Szabó János	9	900
234555	Szilágyi Pál	2	700
123444	Vincze Ildikó	1	800
333333	Kovács István	2	500

Eredmény:

RészlegID	*AVG(Fizetés)*
9	650
1	800

Példák

- *Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500euro, átlagfizetés szerint növekvő sorrendben!*

```
SELECT ReszlegID, AVG(Fizetes)
FROM Alkalmazottak
GROUP BY ReszlegID
HAVING AVG(Fizetes)>500
ORDER BY AVG(Fizetes)
```

- ORDER BY nélkül – *Mi szerint lettek volna rendezve a sorok az eredmény relációban?*

Példák

- *Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500euro, átlagfizetés szerint növekvő sorrendben!*

```
SELECT ReszlegID, AVG(Fizetes)
FROM Alkalmazottak
GROUP BY ReszlegID
HAVING AVG(Fizetes)>500
ORDER BY AVG(Fizetes)
```

- ORDER BY nélkül – Mi szerint lettek volna rendezve a sorok az eredmény relációban? *– A csoportosító attribútum értékei alapján.*

Példák

A)

```
SELECT ReszlegID, AVG(Fizetes)
FROM Alkalmazottak
WHERE AVG(Fizetes)>500
GROUP BY ReszlegID
```

B)

```
SELECT MAX(AVG(Fizetes))
FROM Tanarok
GROUP BY TanszekKod
```

C)

```
SELECT ReszlegID, AVG(Fizetes) Atlag
FROM Alkalmazottak
GROUP BY ReszlegID
HAVING Atlag>500
```

Példák

A)

```
SELECT ReszlegID, AVG(Fizetes)
FROM Alkalmazottak
WHERE AVG(Fizetes)>500
GROUP BY ReszlegID
```

B)

```
SELECT MAX(AVG(Fizetes))
FROM Tanarok
GROUP BY TanszekKod
```

C)

```
SELECT ReszlegID, AVG(Fizetes) Atlag
FROM Alkalmazottak
GROUP BY ReszlegID
HAVING Atlag>500
```

Összesítő függvény másodnevének használata

- *Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500euro, átlagfizetés szerint növekvő sorrendben!*

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

- Az összesítő függvénynek adott másodnév az ORDER BY záradékban használható, a HAVING záradékban viszont NEM.

Lekérdezés végrehajtása-általános eset

a) Az operandus relációkkal a WHERE feltételét figyelembe véve elvégezzük a join-t (esetleg a Descartes szorzat műveletét). *(a1) JOIN, a2) WHERE)*

b) Az így kapott eredmény relációra alkalmazzuk a csoportosítást.

c) Töröljük azon sorokat, melyek nem tesznek eleget a csoportosítási feltételnek.

d) Töröljük az oszlopokat, amelyek nem szerepelnek a céllistában.

e) Ha van megadva rendezési feltétel, e szerint rendezzük az eredményreláció rekordjait.

Feladatok

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

Q7) Adjuk meg azon turisták nevét, akik legalább kétszer foglaltak szállást (valamelyik szálláshelyen)!

Q8) Adjuk meg a magyarországi helységeket, ahol a szállások átlagára éjszakánként nagyobb, mint 50 euro!

Q9) Hogyan módosul a Q7) lekérdezés, ha legalább két különböző szálláshelyen foglalást végzett turisták érdekelnek?

Feladatok

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

Q7) Adjuk meg azon turisták nevét, akik legalább kétszer foglaltak szállást (valamelyik szálláshelyen)!

Q8) Adjuk meg a magyarországi helységeket, ahol a szállások átlagára éjszakánként nagyobb, mint 50 euro!

Q9) Hogyan módosul a Q7) lekérdezés, ha legalább két különböző szálláshelyen foglalást végzett turisták érdekelnek?

↳ *COUNT(DISTINCT SzID)*

Q10) Adjuk meg a legdrágább szállás nevét!

Feladatok

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

Q7) Adjuk meg azon turisták nevét, akik legalább kétszer foglaltak szállást (valamelyik szálláshelyen)!

Q8) Adjuk meg a magyarországi helységeket, ahol a szállások átlagára éjszakánként nagyobb, mint 50 euro!

Q9) Hogyan módosul a Q7) lekérdezés, ha legalább két különböző szálláshelyen foglalást végzett turisták érdekelnek?

↳ *COUNT(DISTINCT SzID)*

Q10) Adjuk meg a legdrágább szállás nevét! ← *alkérdés*

Alkérdések

- Az SQL nyelv lehetőséget biztosít arra, hogy a lekérdezések feltételében is használhassunk SQL lekérdező parancsot. Ilyenkor a külső és a belső SELECT parancsot megkülönböztetjük egymástól, és az ilyen jellegű lekérdezéseket beágyazott lekérdezéseknek (=alkérdésnek) nevezzük.
- Főkérdés – a legkülső szintű kérdés, amely alkérdést (beágyazott kérdést) tartalmaz.
- Egyszerű alkérdés – olyan alkérdés, mely csak egyszer kerül kiértékelésre.
- Korrelált kérdés – olyan alkérdés, amely többször is kiértékelődik.

Alkérdések

- Alkérdést tartalmazó SQL parancs általános formája:

```
SELECT <attribútum_lista>
FROM <tábla>
WHERE <kifejezés> <operátor>
    (SELECT <attribútum_lista>
     FROM <tábla>);
```

- A rendszer először az alkérdést hajtja végre és annak eredményét használja a „fő” lekérdezés; kivétel: korrelált alkérdések.
- A kereskedelmi rendszerek különböző mélységig tudják az alkérdéseket kezelni (*MS SQL: max. mélység – 32*).

Alkérdések csoportosítása

- Skalár értéket visszaadó alkérdés (single row)
  - egy oszlop, egy sor (=, <>, <= , >=)
- Többsoros alkérdés (multiple-row subquery)
  - egy oszlop, több sor
    - Tartalmazás vizsgálata: IN, NOT IN
    - Alkérdés valamely vagy minden sorának vizsgálata: ALL, ANY|SOME
  - egy vagy több oszlop, több sor
    - Alkérdés üresség̋ének vizsgálata: EXISTS, NOT EXISTS

Skalár értéket visszaadó alkérdések

- A belső lekérdezés egyetlen értéket szolgáltat, mely konstanként használható.
- Az eredményt egy attribútummal vagy egy másik konstanssal összehasonlíthatjuk.
- **Vigyázat!** Abban az esetben, ha az alkérdés nulla vagy egynél több sort eredményez, a lekérdezés futás közbeni hibát fog jelezni. Ugyancsak hibajelzést kapunk, ha egynél több oszlopa lesz az alkérdésbeli eredmény relációnak.
- A skalár értéket adó alkérdéssel használható operátorok:

  =, <, <=, >, >=, <>.

Példa

Részlegek(RészlegID,Név,Helység,*ManSzemSzám*);

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

- *Keressük a „Tervezés” nevű részleg managerének a nevét!*

Példa

Részlegek(RészlegID,Név,Helység,*ManSzemSzám*);

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

- *Keressük a „Tervezés” nevű részleg managerének a nevét!*

```
SELECT Név
FROM Alkalmazottak
WHERE SzemSzám =
   (SELECT ManSzemSzám
    FROM Részlegek
    WHERE Név = 'Tervezés');
```

- **Fontos:** Egyetlen „Tervezés” nevű részleg lehet az adatbázisban. ↔ UNIQUE megszorítás a *Név* mezőre.

Példa

- *Keressük a „Tervezés” nevű részleg managerének a nevét!*

```
SELECT Név
FROM Alkalmazottak
WHERE SzemSzám =
   (SELECT ManSzemSzám
    FROM Részlegek
    WHERE Név = 'Tervezés');
```

- A lekérdezés processzor először az alkérdést értékeli ki, ennek eredményeként egy skalár értéket (pl. 123444) kapunk. A fő lekérdezés ezzel a skalár értékkel fog dolgozni:

```
SELECT Név
FROM Alkalmazottak
WHERE SzemSzám = 123444
```

Skalár értéket visszaadó alkérdések

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

- **Példa:** *Keressük azon részlegeket és az alkalmazottak minimális fizetését minden részleg esetén, ahol a minimális fizetés nagyobb, mint a 2-es ID-jú részlegen a maximális fizetés.*

Skalár értéket visszaadó alkérdések

- A HAVING záradékban is szerepelhet alkérdés.

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

- **Példa:** *Keressük azon részlegeket és az alkalmazottak minimális fizetését minden részleg esetén, ahol a minimális fizetés nagyobb, mint a 2-es ID-jú részlegen a maximális fizetés.*

```
SELECT RészlegID, MIN(Fizetés)
FROM Alkalmazottak
GROUP BY RészlegID
HAVING MIN(Fizetés) >
          (SELECT MAX(Fizetés)
           FROM Alkalmazottak
           WHERE RészlegID = 2)
```

Példa

A)

```
SELECT SzemSzám, Név
FROM Alkalmazottak
WHERE Fizetés = MIN(Fizetés)
```

B)

```
SELECT SzemSzám, Név
FROM Alkalmazottak
WHERE Fizetés =
   (SELECT MIN(Fizetés)
    FROM Alkalmazottak
    GROUP BY RészlegID);
```

Példa

A)

```
SELECT SzemSzám, Név
FROM Alkalmazottak
WHERE Fizetés = MIN(Fizetés)
```

- Vigyázat: SQL standard szerint NEM megfelelő!

B)

```
SELECT SzemSzám, Név
FROM Alkalmazottak
WHERE Fizetés =
   (SELECT MIN(Fizetés)
    FROM Alkalmazottak
    GROUP BY RészlegID);
```

- Csínján bánjunk a csoportosítással!

Többsoros alkérdések

- A belső lekérdezés egyoszlopos relációt szolgáltat.
- Többsoros alkérdés esetén a fő lekérdezés WHERE záradékának feltétele olyan operátorokat tartalmazhat, amelyeket egy - a belső lekérdezés által visszaadott - *R* relációra alkalmazhatunk ↔ az eredmény logikai érték lesz.
  - **EXISTS** *R* – feltétel, mely akkor és csak akkor igaz, ha *R* nem üres (**EXISTS** $R \Leftrightarrow R \neq \emptyset$).
  - *Analóg módon:* **NOT EXISTS** *R* $\Leftrightarrow R = \emptyset$

Példa

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

Managerek(*SzemSzám*)

- *Mit fog kiírni?*

```
SELECT Név
FROM Alkalmazottak a, Managerek m
WHERE a.SzemSzám = m.SzemSzám
   AND EXISTS
        (SELECT *
         FROM Alkalmazottak
         WHERE Fizetés > 500)
```

Példa

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

Managerek(*SzemSzám*)

- *Mit fog kiírni?*

```
SELECT Név
FROM Alkalmazottak a, Managerek m
WHERE a.SzemSzám = m.SzemSzám
   AND EXISTS
        (SELECT *
         FROM Alkalmazottak
         WHERE Fizetés > 500)
```

- A managerek nevét adja meg, ha van olyan alkalmazott, akinek a fizetése nagyobb, mint 500€.

Példa

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

`Managerek(`*`SzemSzám`*`)`

- *Keressük azon tanárokat, akik még diákok!*

Példa

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

Managerek(*SzemSzám*)

- *Keressük azon tanárokat, akik még diákok!*

```
SELECT Név FROM Tanárok
WHERE EXISTS
   (SELECT Név FROM Diákok
    WHERE Diákok.SzemSzám = Tanárok.SzemSzám);
```

- A feladatot másképp is meg lehet oldani (*lsd. halmazműveletek*).

Tartalmazás vizsgálata

- Bizonyos operátoroknak egy skaláris s értékre is szükségük van.
  - s **IN** *R* – feltétel, mely akkor igaz, ha s egyenlő valamelyik *R*-beli értékkel.
  - s **NOT IN** *R* - igaz, ha s egyetlen *R*-beli értékkel sem egyenlő.
- **Példa:**

```
Áruk(ÁruID, Név, MértEgys, MennyRakt, CsopID);
Szállítók(SzállID, Név, Helység, UtcaSzám);
Szállít(SzállID, ÁruID, Ár);
```

Adjuk meg azon szállítók nevét és címét, amelyek valamilyen csokit szállítanak (Áruk.Név LIKE '%csoki%')!

Tartalmazás vizsgálata

Áruk(ÁruID, Név, MértEgys, MennyRakt, *CsopID*);
Szállítók(SzállID, Név, Helység, UtcaSzám);
Szállít(*SzállID, ÁruID*, Ár);

- **Példa:** *Adjuk meg azon szállítók nevét és címét, amelyek valamilyen csokit szállítanak (Áruk.Név LIKE ‘%csoki%’)*

Tartalmazás vizsgálata

Áruk(ÁruID, Név, MértEgys, MennyRakt, *CsopID*);
Szállítók(SzállID, Név, Helység, UtcaSzám);
Szállít(*SzállID, ÁruID*, Ár);

- **Példa:** *Adjuk meg azon szállítók nevét és címét, amelyek valamilyen csokit szállítanak (Áruk.Név LIKE '%csoki%')*

```
SELECT Név, Helység, UtcaSzám
FROM Szállítók
WHERE SzállID IN
    (SELECT SzállID
     FROM Szállít
     WHERE ÁruID IN
        SELECT ÁruID
        FROM Áruk
        WHERE Név LIKE '#csoki#')
```

Tartalmazás vizsgálata

Áruk(ÁruID, Név, MértEgys, MennyRakt, *CsopID*);
Szállítók(SzállID, Név, Helység, UtcaSzám);
Szállít(*SzállID, ÁruID*, Ár);

- **Példa:** *Adjuk meg azon szállítók nevét és címét, amelyek valamilyen csokit szállítanak (Áruk.Név LIKE '%csoki%')*

```
SELECT Név, Helység, UtcaSzám
FROM Szállítók
WHERE SzállID IN
    (SELECT SzállID
     FROM Szállít
     WHERE ÁruID IN
        SELECT ÁruID
        FROM Áruk
        WHERE Név LIKE '#csoki#')
```

Alkérdés nélküli megoldás:
Áruk, Szállítok, Szállít táblák természetes összekapcsolása ("inner join-ja").

Alkérdés valamely vagy minden sorának vizsgálata

- s <összehasonlító op.> **ALL** R – feltétel, mely akkor igaz, ha $\forall\ t \in R$: (F <összehasonlító op.> t) igaz,
  ahol <összehasonlító op.> lehet: $<,\ \leq,\ >,\ \geq,\ =,\ \neq$

- Az s <> **ALL** R eredménye ugyanaz, mint az s **NOT IN** R feltételé. Ugyanakkor: **(= all) ≡ in**

(5 < **all** {0, 5, 6}) = false

(5 < **all** {6, 10}) = true

(5 = **all** {4, 5}) = false

(5 ≠ **all** {4, 6}) = true (mivel 5 ≠ 4 és 5 ≠ 6)

Alkérdés valamely vagy minden sorának vizsgálata

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

- **Példa:** Tekintsük az előbbi helytelen lekérdezést:

```
SELECT SzemSzám, Név
FROM Alkalmazottak  WHERE Fizetés =
(SELECT MIN(Fizetés)
   FROM Alkalmazottak  GROUP BY RészlegID);
```

Az előbb helytelen volt az „=” jel miatt.

Alkérdés valamely vagy minden sorának vizsgálata

Az előbb helytelen volt az „=” jel miatt.

- Az alkérdés több sort is visszaad →a főkérdés a „> ALL” operátort alkalmazva a Fizetés oszlop értékét összehasonlítja az összes minimális fizetés értékkel az alkérdésből.
- Azon alkalmazottakat kapjuk meg, akiknek fizetése nagyobb, mint minden részlegen a minimális fizetés.

Alkérdés valamely vagy minden **sorának vizsgálata**

SOME ⟺ ANY

- s <összehasonlító op.> **SOME|ANY** R – feltétel, mely akkor igaz, ha $\exists\, t \in R$: (F <összehasonlító op.> t) igaz, ahol <összehasonlító op.> lehet: $<$, $\leq$, $>$, $\geq$, $=$, $\neq$

- Az s = **SOME|ANY** R eredménye ugyanaz, mint az s **IN** R feltételé.

Ugyanakkor: **(≠ SOME) ≠ NOT IN**

(5 < **SOME|ANY** {0, 5, 6}) = igaz

Így olvasd: 5 < néhány|valamely rekordnál a relációból.

(5 < **SOME|ANY** {0, 5}) = hamis

(5 = **SOME|ANY** {0, 5}) = igaz

(5 ≠ **SOME|ANY** {0, 5}) = igaz (mivel 0 ≠ 5)

Alkérdés valamely vagy minden sorának vizsgálata

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

- **Példa:** *Adjuk meg azon alkalmazottakat, akiknek a fizetése nagyobb, mint néhány alkalmazott fizetése a 4-es részlegből!*

```
SELECT Név
FROM Alkalmazottak
WHERE Fizetés > SOME
          (SELECT Fizetés
           FROM Alkalmazottak
           WHERE RészlegID = 4)
```

Más megoldás?

Alkérdés valamely vagy minden sorának vizsgálata

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

- **Példa:** *Adjuk meg azon alkalmazottakat, akiknek a fizetése nagyobb, mint néhány alkalmazott fizetése a 4-es részlegből!*

  JOIN-nal:

```
SELECT Név
FROM Alkalmazottak a1, Alkalmazottak a2
WHERE a1.Fizetés > a2.Fizetés
      AND a2.RészlegID=4
```

Feladatok

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

Q11) Adjuk meg azon szállásokat (helységnévvel együtt), amelyek egyedül rendelkeznek 5csillaggal a helységükben (vagyis: nincs más 5 csillagos szállás az adott helységben)!

Feladatok

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

Q11) Adjuk meg azon szállásokat (helységnévvel együtt), amelyek egyedül rendelkeznek 5csillaggal a helységükben (vagyis: nincs más 5 csillagos szállás az adott helységben)!

↳ *korrelált alkérdés*

Korrelált alkérdések

- Korrelált alkérdés – olyan alkérdés, amely a fő lekérdés által kiválasztott minden sorra újra végrehajtódik.
  - Az alkérdés végrehajtásának a kimenete az adott sortól függ.
  - Az alkérdés többszöri kiértékelését egy, az alkérdésen kívüli sorváltozóval érjük el.
- A korrelált lekérdezések használata közben figyelembe kell vennünk a nevek érvényességi körére vonatkozó szabályokat.
  - Egy lekérdezésben szereplő attribútum a FROM záradékban szereplő valamely sorváltozóhoz tartozik, ha a sorváltozó relációja tartalmazza az attribútumot. Ellenkező esetben: a lekérdezést „körbeölelő” magasabb rendű lekérdezéseket kell vizsgálnunk.
- Megj. *Nem ajánlott a használatuk*, de vannak esetek, amikor nem lehet csak lekérdezések használatával a feladatot másként megoldani.

Feladat

Helysegek(HKod, HNev, *OKod*)
Szallasok(SzID,SzNev,*Hkod,SztID,Csillag,Ar/Ej*)

Q11) Adjuk meg azon szállásokat (helységnévvel együtt), amelyek egyedül rendelkeznek 5csillaggal a helységükben (vagyis: nincs más 5 csillagos szállás az adott helységben)!

```
SELECT SzNev, HNev
FROM Szallasok sz1, Helysegek h
WHERE sz1.Hkod = h.Hkod AND Csillag = 5
  AND NOT EXISTS (
    SELECT sz2.SzID
    FROM Szallasok sz2
    WHERE sz2.Csillag=5
      AND sz2.Hkod=sz1.Hkod
      AND sz2.SzID<>sz1.SzID)
```

Lekérdezés kiértékelése

```
(1) SELECT SzNev, HNev
(2) FROM Szallasok Sz1,
(3)  Helysegek h
(4) WHERE sz1.Hkod = h.Hkod
(5)  AND Csillag = 5
(6)  AND NOT EXISTS (
(7)   SELECT sz2.SzID
(8)   FROM Szallasok sz2
(9)   WHERE sz2.Csillag = 5
(10)   AND sz2.Hkod = sz1.Hkod
(11)   AND sz2.SzID<>sz1.SzID)
```

- Sz1 sorváltozó: végigjárja a Szallasok relációt;
- Minden sorra az Sz1-ből az Sz2 sorváltozó végigjárja a Szallasok relációt;
- Legyen sz1 egy sor a Szallasok relációból, amelyet a fölekérdezés az eredménybe helyez, ha megfelel a fölekérdezésben szereplő feltételeknek:
  - A 6.sorban szereplő feltétel akkor lesz igaz, ha az alkérdés üres halmazt eredményez.

Alkérdések a FROM záradékban

- A FROM záradékban a tényleges reláció(k) listájában előfordulhatnak alkérdések is.
- Ezeket zárójelek között adhatjuk meg.
- Vigyázat! Az alkérdés eredmény relációjának nincs neve $\Longrightarrow$másodnévadás szükséges.
- A FROM záradékban kiértékelődik az alkérdés, így az alkérdés által létrehozott reláció soraira már ugyanúgy tudunk hivatkozni, mint a FROM lista többi relációjának soraira.

Példa

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

- *Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500 €!*

1. megoldás (ismét): (HAVING-gel)

```
SELECT ReszlegID, AVG(Fizetes)
FROM Alkalmazottak
 GROUP BY ReszlegID
 HAVING AVG(Fizetes)>500
```

Példa

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

- *Keressük azon részlegeket, ahol az alkalmazottak átlagfizetése nagyobb, mint 500 €!*

2.megoldás: HAVING nélkül, alkérdéssel

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

- **Megj.** A HAVING záradék kiküszöbölhető lett alkérdéssel: az ReszlegAtlag(Atlag) temporális reláció meghatározása után az attribútumai használhatóak a WHERE záradékban.

Feladatok

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

Q10) Adjuk meg a legdrágább szállás nevét!

Q12) Adjuk meg a kolozsvári turisták nevét, akik foglaltak le szállást párizsi 5 csillagos szállodákban!

Q13) Adjuk meg a magyarországi helységeket, ahol a szállások átlagára éjszakánként nagyobb, mint 50 euro!

Q14) Adjuk meg azon helységeket, ahol a legtöbb szállás található!

Feladatok

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

Q15) Keressük azon helységeket, ahol minden szállástípusból található szállás!

Q16) Keressük azon helységeket, ahol legalább annyi szállástípus található, mint Kolozsváron!

Q17) Mely turisták foglaltak Párizsban szállodát ÉS Budapesten panziót?

Feladatok

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

Q15) Keressük azon helységeket, ahol minden szállástípusból található szállás! ← *rel.algebrában: osztás*

Q16) Keressük azon helységeket, ahol legalább annyi szállástípus található, mint Kolozsváron! ← *rel.algebrában: osztás*

Q17) Mely turisták foglaltak Párizsban szállodát ÉS Budapesten panziót? ← *halmazművelet*

Halmazműveletek SQL-ben

- A relációs algebra halmazműveleteit (egyesítés, metszet és különbség) használhatjuk az SQL nyelvben, azzal a feltétellel, hogy az operandus relációknak ugyanaz legyen az attribútumhalmaza.
- A megfelelő kulcsszavak:
  - ∪ ↔ `UNION`
  - ∩ ↔ `INTERSECT`
  - − ↔ `EXCEPT`
- Ha a kereskedelmi rendszer nem támogatja az INTERSECT [EXCEPT] műveletet (pl. MySQL), alkalmazhatjuk a [NOT] EXISTS vagy [NOT] IN záradékokat.

Példa halmazműveletre

Szállítók(SzállID, Név, Helység, UtcaSzám);

Vevők(VevőID,Név,Helység,UtcaSzám,Mérleg,Hihetőség)

- *Keressük azon kolozsvári cégeket, amelyekkel kapcsolatban áll a cégünk (lehet vevő vagy szállító).*

Példa halmazműveletre

Szállítók(SzállID, Név, Helység, UtcaSzám);

Vevők(VevőID,Név,Helység,UtcaSzám,Mérleg,Hihetőség)

- *Keressük azon kolozsvári cégeket, amelyekkel kapcsolatban áll a cégünk (lehet vevő vagy szállító).* ← egyesítés

```
(SELECT Név, UtcaSzám
FROM Szállítók
WHERE Helység = 'Kolozsvár')
UNION
(SELECT Név, UtcaSzám
FROM Vevők
WHERE Helység = 'Kolozsvár')
```

Példa halmazműveletre

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

`Managerek(`*`SzemSzám`*`);`

- *Keressük azon alkalmazottakat, akik NEM managerek.*

Példa halmazműveletre

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

Managerek(*SzemSzám*);

- *Keressük azon alkalmazottakat, akik NEM managerek.*

↳ különbség

```
(SELECT SzemSzám, Név
 FROM Alkalmazottak)
EXCEPT
(SELECT SzemSzám, Név
 FROM Managerek m, Alkalmazottak a
 WHERE m.SzemSzám = a.SzemSzám);
```

Példa halmazműveletre

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

Managerek(*SzemSzám*);

- *Keressük azon alkalmazottakat, akik NEM managerek.*

↳ különbség

- Más megoldás (pl. ha a kereskedelmi rendszer nem támogatja az EXCEPT műveletet):

```
(SELECT SzemSzám, Név
 FROM Alkalmazottak
 WHERE SzemSzám NOT IN
   (SELECT SzemSzám
    FROM Managerek m, Alkalmazottak a
    WHERE m.SzemSzám = a.SzemSzám);
```

Példa halmazműveletre

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

Managerek(*SzemSzám*);

- *Keressük azon alkalmazottakat, akik NEM managerek.*

↳ különbség

- Más megoldás (pl. ha a kereskedelmi rendszer nem támogatja az EXCEPT műveletet):

```
(SELECT SzemSzám, Név
 FROM Alkalmazottak a
 WHERE NOT EXISTS
   (SELECT a.SzemSzám
    FROM Managerek m
    WHERE m.SzemSzám = a.SzemSzám);
```

Példa halmazműveletre

Diákok (BeiktatásiSzám, Név, SzemSzám, Cím,
SzületésiDatum, *CsopKod*, Átlag);

Tanárok (TanárKod, Név, SzemSzám, Cím, PhD,
TanszékKod, *BeosztásKod*, Fizetés);

- Tekintsük az alkérdések témakörnél tárgyalt egyik példát:
  Keressük azon tanárokat, akik még diákok!

Példa halmazműveletre

```
Diákok (BeiktatásiSzám, Név, SzemSzám, Cím,
        SzületésiDatum, CsopKod, Átlag);
Tanárok (TanárKod, Név, SzemSzám, Cím, PhD,
        TanszékKod, BeosztásKod, Fizetés);
```

- Tekintsük az alkérdések témakörnél tárgyalt egyik példát:
  Keressük azon tanárokat, akik még diákok! ← metszet

```
SELECT Név FROM Tanárok
WHERE EXISTS
 (SELECT Név FROM Diákok
WHERE Diákok.SzemSzám = Tanárok.SzemSzám);
```

  - Ez a megoldás alkalmazható, ha a kereskedelmi rendszer nem támogatja az INTERSECT műveletet.

Példa halmazműveletre

```
Diákok (BeiktatásiSzám, Név, SzemSzám, Cím,
        SzületésiDatum, CsopKod, Átlag);
Tanárok (TanárKod, Név, SzemSzám, Cím, PhD,
        TanszékKod, BeosztásKod, Fizetés);
```

- *Keressük azon tanárokat, akik még diákok!*
  - Megoldás INTERSECT-tel:

```
(SELECT Név FROM Tanárok)
INTERSECT
(SELECT Név FROM Diákok);
```

Ismétlődő sorok

- Különbség az SQL nyelv relációi és az absztrakt módon definiált relációk között: az SQL-ben a relációkat nem halmazoknak tekintjük, hanem multihalmazoknak. Vagyis:
  - A SELECT parancs eredményében szerepelhet két vagy több teljesen azonos sor.
- Van lehetőség ezen ismétlődések megszüntetésére: a DISTINCT kulcsszó megadása a SELECT után.

Pl.
```
SELECT DISTINCT Név
FROM Diákok
```

Ismétlődő sorok

- A SELECT paranccsal ellentétben a halmazműveletek (UNION, EXCEPT és INTERSECT) megszüntetik az ismétlődéseket.
- Ha nem szeretnénk, hogy az ismétlődő sorok eltűnjenek, a műveletet kifejező kulcsszó után az ALL kulcsszót kell használjuk. **Megj.** MSSQL-ben csak UNION-nal működik.
- **Példa:** *Keressük azokat a személyeket, akik lehetnek tanárok vagy diákok!*

```
(SELECT Név FROM Diákok)
    UNION ALL
(SELECT Név FROM Tanárok)
```

Feladatok

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

Q17) Mely turisták foglaltak Párizsban szállodát ÉS Budapesten panziót?

Q18) Adjuk meg azokat a helységeket, ahova a budapesti VAGY a bukaresti turisták (vagy is-is) foglaltak szállást!

Q19) Mely turisták foglaltak CSAK Párizsban szállást?

Q20) Melyik az a szállástípus, amely NEM található meg Kolozsváron (megj.:attól még máshol igen)?

Feladatok

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

Q21) Adjuk meg minden helység esetén a szállások típusainak számát (azon helységeket is, ahol egyetlen szálláshely sem található)!

Q21') Melyik helységben nincs egy szálláshely sem? – OUTER JOIN-nal?

Q22) Adjuk meg azon szálláshelyeket, ahol történt legalább két foglalás az elmúlt hónapban vagy Kolozsváron találhatóak!