

Adatbázis kezelés I.

SQL bevezető

Rostagni Csaba

2022.08.28

Ezen az órán... I

- 1 Bevezető
- 2 Egyszerű lekérdezések (SELECT)
- 3 Műveleti jelek
- 4 Feltételes lekérdezések (WHERE)
- 5 A NULL érték
- 6 Rendezés (ORDER BY)
- 7 Sorok számának limitálása (LIMIT)

Ezen az órán... II

- 8 Számított mezők
- 9 Matematikai függvények
- 10 Szöveg függvények
- 11 Dátum és idő függvények
- 12 Egyéb hasznos függvények
- 13 Aggregált (összesítő) függvények
- 14 Csoportosítás (GROUP BY)

Ezen az órán... III

15 Feltételek csoportosított mezőre (HAVING)

16 Többtáblás lekérdezések

17 Típusok

18 Adatbázis létrehozása és törlése

19 Tábla létrehozása és törlése

20 Adatok beszúrása és törlése

21 Anomáliák

Ezen az órán... IV

- 22 Kulcsok
- 23 Normalizáció
- 24 Adatbázis módosítása
- 25 Táblák módosítása
- 26 Adatok módosítása (UPDATE)
- 27 Megszorítások
- 28 Index

Ezen az órán... V

- 29 Kapcsolatok típusai
- 30 2NF - A második normálforma
- 31 3NF - A harmadik normálforma
- 32 Mi az az EK model?
- 33 EK diagram elemei
- 34 EK diagram példák
- 35 ER modell leképezése

Tartalom I

1 Bevezető



Speciális karakterek

A diákon különböző speciális jelek találhatók meg.

aposztróf

- A MySQL aposztóffal jelöli a szöveget.

pl.: 'szoveg'

-  + 

backtick

- Így jelöli adatbázisokat, táblákat és a mezőneveket.

pl.: `tablanev`

-  + 

Megjegyzés (egy soros)

- A # karaktertől a sor végéig
- A --_ karaktersorozattól a sor végéig
 - Utána egy szóköz kell!
pl.: --_Ez_egy_megjegyzés
 - Kissé eltér a szabványtól, de **ezt használjuk!**

Linkek:

- MySQL dokumentáció: Megjegyzések
- MySQL dokumentáció: A szabványtól eltérő megjegyzés

Megjegyzés (több soros)

- Nyitó karaktersorozat: `/*`
- Záró karaktersorozat: `*/`
- Szöveg közben használható
pl.: `/* Ez szövegeközi vagy több soros megjegyzes */`

Linkek:

- MySQL dokumentáció: Megjegyzések
- MySQL dokumentáció: A szabványtól eltérő megjegyzés

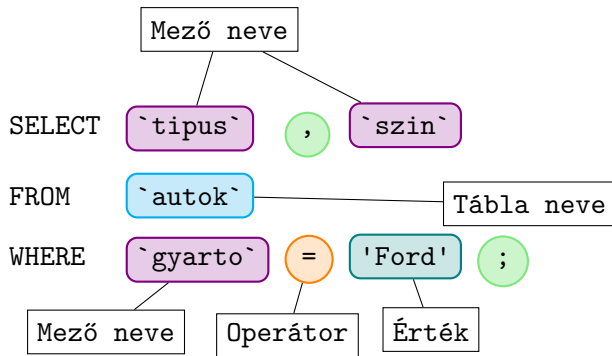
Különböző típusú értékek jelölése

- A szöveges értékeket aposztróf közé kell elhelyezni.
`'Nagy Lajos'`
- A dátumokat és az időket is aposztróf között fogjuk szabványosan megadni.
`'2018-12-01'`
`'15:55:30'`
`'2018-12-01 12:55:30'`
- Az egész számokat számmal megadhatjuk a szokásos módon.
`10`
- A valós számokat **tizedes ponttal** kell elválasztani!
`10.5`

Tartalom I

2 Egyszerű lekérdezések (SELECT)

Egyszerű SELECT felépítése



Első lekérdezésünk

Listázzunk ki minden adatot az **autok** táblából!

MySQL

```
SELECT *  
FROM `autok`;
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| ABC-123 | Ford | Focus | M1 | diesel | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

Mezők kiválasztása

Listázzuk ki az összes autó **rendszámát** és **típusát**!

MySQL

```
SELECT `rendszám`, `típus`  
FROM `autok`;
```

| rendszám | típus |
|------------|--------|
| XXX-111 | Adam |
| AAA-555 | Jazz |
| ABC-123 | Focus |
| AA-AX-1234 | Fiesta |

Tartalom I

- 3 Műveleti jelek
 - Összehasonlító operátorok
 - Logikai operátorok

Tartalom

- 3 Műveleti jelek
 - Összehasonlító operátorok
 - Logikai operátorok

Összehasonlító operátorok

| SQL | Mat. | Megnevezés |
|-----|------|-----------------|
| < | < | kisebb |
| > | > | nagyobb |
| <= | ≤ | kisebb egyenlő |
| >= | ≥ | nagyobb egyenlő |
| = | = | egyenlő |
| <> | ≠ | nem egyenlő |

táblázat: összehasonlító operátorok

Megjegyzés:

- A != is a nem egyenlőt jelenti (MySQL, MS SQL, Oracle, ...)
- A ^= is a nem egyenlőt jelenti (Oracle)
- A <> formátum felel meg az ANSI szabványnak, ez az elvárt!

Tartalom

- 3 Műveleti jelek
 - Összehasonlító operátorok
 - Logikai operátorok

Logikai operátorok

| SQL | Mat. | Megnevezés |
|-----|----------|------------|
| NOT | \neg | nem |
| AND | \wedge | és |
| OR | \vee | vagy |

táblázat: logikai operátorok

Az operátorok precedencia szerint csökkenő sorrendben vannak feltüntetve.

Linkek:

- MySQL dokumentáció: Operátorok precedenciája

Tartalom I

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Feltételek megadása (WHERE)

Listázzuk ki a **benzines** autók minden adatát

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin';
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

Megadott mezők és feltételek

Listázzuk ki a **benzines** autók **gyártóját** és **típusát**!

MySQL

```
SELECT `gyarto`,`tipus`  
FROM `autok`  
WHERE `uzemanyag` = 'benzin';
```

| gyarto | tipus |
|--------|--------|
| Opel | Adam |
| Ford | Fiesta |

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- **Tagadás**
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Egyszerű tagadás

Listázzuk ki a **nem benzines** autókat.

MySQL

```
SELECT * FROM `autok`  
WHERE `uzemanyag` <> 'benzin';
```

| rendszer | gyarto | tipus | kategoria | uzemanyag | szin |
|----------|--------|-------|-----------|-----------|------|
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| ABC-123 | Ford | Focus | M1 | diesel | kék |

- Használhattuk volna a `!=` operátort, csak az SQL szabvány nem azt tartalmazza.
- Egyszerű feltételek esetén ilyen egyszerű tagadni.

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Logikai operátorok: ÉS (AND)

Listázzuk ki a **benzines Fordok** minden adatát!

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
      AND `gyarto` = 'Ford';
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

Logikai operátorok: VAGY (OR)

Listázzuk ki a **piros** vagy **sárga** színű autók minden adatát!

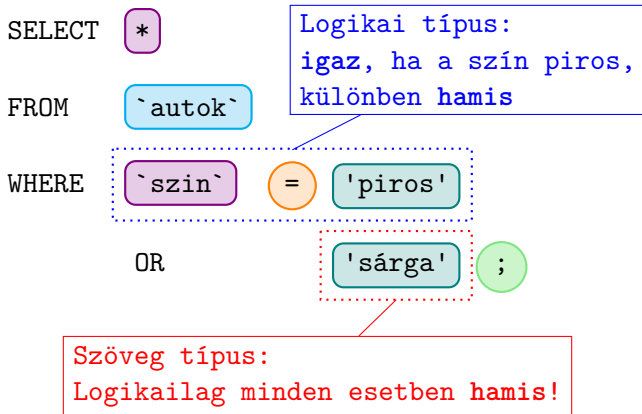
```
SELECT *  
FROM autok  
WHERE szin = 'piros'  
       OR 'sárga';
```

logikai hiba

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|----------|--------|-------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |

- Hol van a sárga autó?

Logikai operátorok: VAGY (OR)



- Mivel a második feltétel mindig hamis, így olyan, mintha ott se lenne.

Logikai operátorok: VAGY (OR)

Listázzuk ki a **piros**, vagy **sárga** színű autók minden adatát!

MySQL

```
SELECT *  
FROM `autok`  
WHERE `szin` = 'piros'  
      OR `szin` = 'sárga';
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

Logikai operátorok: VAGY (OR)

Listázzuk ki a **hibrid** és a **benzines** autók minden adatát!

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
      OR `uzemanyag` = 'hibrid';
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

- A köznyelvben használt "és" logikailag lehet, hogy "**vagy**"-ot jelent.

És / Vagy operátorok

- `&&`
 - a MySQL egy nem szabványos kiegészítése,
 - a 8.0.17-es verziótól kezdve elavult, meg fog szűnni
- `||`
 - a MySQL egy nem szabványos kiegészítése,
 - a 8.0.17-es verziótól kezdve elavult, meg fog szűnni
 - Az ANSI szabvány szerint összefűzést jelent

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- **ÉS/VAGY kombinálása**
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Logikai operátorok kombinálása (hibás próbálkozás)

Listázzuk ki azoknak a **benzines** autóknak minden adatát, melyek **Ford** vagy **Honda** gyártmányúak.

```
SELECT * FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
    AND `gyarto` = 'Ford'  
    OR `gyarto` = 'Honda';
```

logikai hiba

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

- **Probléma:** a Honda Jazz is megjelent, ami nem benzines!

Logikai operátorok kombinálása hiba magyarázata

Az ÉS logikai operátor precedenciája magasabb.
Elsőnek ez a feltétel teljesül.

SELECT

*

FROM

`autok`

WHERE

`üzemanyag`

=

'benzin'

AND

`gyarto`

=

'Ford'

OR

`gyarto`

=

'Honda'

;

A Honda esetén az üzemanyagra nem szűr.

Logikai operátorok kombinálása

Listázzuk ki azoknak a **benzines** autóknak minden adatát, melyek **Ford** vagy **Honda** gyártmányúak.

MySQL

```
SELECT *  
FROM `autok`  
WHERE (`uzemanyag` = 'benzin')  
      AND (`gyarto` = 'Ford' OR `gyarto` = 'Honda');
```

| rendszám | gyarto | tipus | kategoria | uzemanyag |
|------------|--------|--------|-----------|-----------|
| AA-AX-1234 | Ford | Fiesta | M1 | benzin |

- Megfelelően zárójelezve rövid, tömör kódot kapunk
- Későbbi módosításkor is csak egy helyen kell módosítani

Egyszerű tagadás (NOT)

Listázzuk ki a **nem benzines** autókat.

MySQL

```
SELECT * FROM `autok`  
WHERE NOT ( `uzemanyag` = 'benzin' );
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|----------|--------|-------|-----------|-----------|------|
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| ABC-123 | Ford | Focus | M1 | diesel | kék |

Megjegyzés:

- Ennél az egyszerű példánál a zárójelezés elhagyható, összetettebbeknél célszerű kitenni

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Választás több elem közül

Listázzuk ki a **benzines**, **diesel** és **elektromos** autók minden adatát

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
           OR 'diesel'  
           OR 'elektromos';
```

logikai hiba

- Az ``uzemanyag` = 'benzin'` logikai értéke lehet igaz vagy hamis.
- A `'diesel'` és az `'elektromos'` logikai értéke MINDIG HAMIS!
- Mivel a **(bármilyen) vagy hamis** értéke **(bármilyen)**, így a benzines autók összes adatát kapjuk meg.!

Választás több elem közül

Listázzuk ki a **benzines**, **diesel** és **elektromos** autók minden adatát

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag` = 'benzin'  
      OR `uzemanyag` = 'diesel'  
      OR `uzemanyag` = 'elektromos';
```

- Az `uzemanyag` mezőt fölöslegesen sokszor kellett felsorolni
- ÉS/VAGY együttes alkalmazása okozhat problémát, ha nincs jól zárójelezve

Az IN operátor

Listázzuk ki a **benzines**, **diesel** és **elektromos** autók minden adatát

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag`  
      IN ('benzin', 'diesel', 'elektromos');
```

- Csak egyenlőség vizsgálat esetén alkalmazható
- Az `uzemanyag` mezőt csak egyszer kellett felsorolni
- ÉS alkalmazásakor nem zavar be a precedencia
- Számok esetén az aposztrófok elhagyhatóak. pl.: **IN**(1,5,8)

Linkek:

- MySQL dokumentáció: Az IN operátor

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Azonos mezőre több lehetséges érték tagadással

Listázzuk ki azokat autókat, melyekre nem igaz, hogy **diesel** vagy **benzin** az üzemanyaga.

```
SELECT * FROM `autok`  
WHERE `uzemanyag` <> 'benzin'  
      OR `uzemanyag` <> 'diesel';
```

logikai hiba

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| ABC-123 | Ford | Focus | M1 | diesel | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

- Az `uzemanyag` <> 'benzin' megjeleníti az összes NEM benzinest, így az összes dieselt is!
- Az `uzemanyag` <> 'diesel' megjeleníti az összes NEM dieselt!

Azonos mezőre több lehetséges érték tagadással

Listázzuk ki azokat autókat, melyekre nem igaz, hogy **diesel** vagy **benzin** az üzemanyaga.

MySQL

```
SELECT * FROM `autok`  
WHERE NOT (`uzemanyag` = 'benzin'  
          OR `uzemanyag` = 'diesel');
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|----------|--------|-------|-----------|-----------|------|
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |

- Megnézi minden egyes sorra, hogy diesel vagy benzin üzemű az autó,
- Amennyiben a válasz nem, akkor jeleníti csak meg.

Azonos mezőre több lehetséges érték tagadással

Listázzuk ki azokat autókat, melyekre nem igaz, hogy **diesel** vagy **benzin** az üzemanyaga.

MySQL

```
SELECT * FROM `autok`  
WHERE `uzemanyag` <> 'benzin'  
AND `uzemanyag` <> 'diesel';
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|----------|--------|-------|-----------|-----------|------|
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |

- A feladatot átfogalmazva adódik egy másik megoldás:
 - "Listázzuk ki a se nem **benzines**, se nem **diesel** autók összes adatát."
- A zárójel felbontásakor a De Morgan-azonosságokat figyelembe kell venni

Linkek:

- De Morgan-azonosságok - Wikipedia

A NOT IN operátor

Listázzuk ki a se nem **benzines**, se nem **diesel** autók összes adatát.

MySQL

```
SELECT *  
FROM `autok`  
WHERE `uzemanyag`  
      NOT IN ('benzin', 'diesel');
```

- Egy zárójelben felsorolhatjuk a kizárandó értékeket.

Linkek:

- [MySQL dokumentáció: A NOT IN operátor](#)

Tartalom

4 Feltételes lekérdezések (WHERE)

- Egyszerű feltételek megadása
- Tagadás
- ÉS/VAGY alkalmazása
- ÉS/VAGY kombinálása
- Azonos mezőre több lehetséges érték (IN)
- Azonos mezőre több lehetséges érték tagadással (NOT IN)
- Két érték között (BETWEEN)

Két érték közötti vizsgálat

Jelenítsük meg a 10 és 20 év közötti diákokat.

MySQL

```
SELECT * FROM `diakok`  
WHERE   `kor` >= 10  
      AND `kor` <= 20;
```

- A `kor` mezőt kétszer is szerepeltetni kell!
- Oda kell figyelni, hogy egyik oldalt se maradjon le az egyenlőség!
- Oda kell figyelni, hogy **és** kapcsolat legyen a két feltétel között.
- Összetett feltételben ez okozhat gondot!

A BETWEEN ... AND ... operátor

Jelenítsük meg a 10 és 20 év közötti diákokat.

```
SELECT * FROM `diakok`  
WHERE `kor` BETWEEN 10 AND 20;
```

MySQL

- A 10 és 20, azaz a minimum és a maximum is benne lesz a szűrésben, nem lehet lefelejtetni az egyenlőséget
- Egy egységet alkot, így összetett feltételben zárójelek nélkül is használható

Linkek:

- [MySQL dokumentáció: A BETWEEN operátor](#)

A NOT BETWEEN ... AND ... operátor

Jelenítsük meg azokat a diákokat, akik élettkora nem esik 10 és 20 közé.

MySQL

```
SELECT * FROM `diakok`  
WHERE `kor` NOT BETWEEN 10 AND 20;
```

Ugyanígy működne, ha a sima BETWEEN eredményét letagadnánk.

MySQL

```
SELECT * FROM `diakok`  
WHERE NOT (`kor` BETWEEN 10 AND 20);
```

- A zárójel itt elhagyható, az átláthatóság miatt került be.

Linkek:

- [MySQL dokumentáció: A NOT BETWEEN operátor](#)

Tartalom I

5 A NULL érték

A NULL érték

- A null érték speciális érték
- A NULL nem egyenlő 0-val!
- A NULL nem egyenlő az üres szöveggel!
- Adatbázisan NULL jelentése: Az adott mező értéke nincs megadva, nincs kitöltve, ismeretlen.

A NULL vizualizálása



ábra: A NULL érték vizualizálása

Forrás: <https://www.reddit.com/r/ProgrammerHumor>

IS NULL

Az IS NULL segítségével ellenőrizhetjük, hogy a mező értéke NULL-e.

```
SELECT * FROM `autok`  
WHERE `tipus` IS NULL;
```

MySQL

IS NOT NULL

Az IS NOT NULL segítségével ellenőrizhetjük, hogy a mező értéke **nem** NULL.

```
FROM `autok`  
WHERE `tipus` IS NOT NULL;
```

MySQL

COALESCE()

```
COALESCE(value,...)
```

- Visszaadja az első **nem** NULL értéket

```
SELECT COALESCE(NULL, NULL, NULL, 'A', 'B') AS `e`  
FROM DUAL;
```

MySQL

e

A

Tartalom I

6 Rendezés (ORDER BY)

ORDER BY

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto`;
```

- Az **ORDER BY** után sorolható fel, hogy melyik mező(k) alapján, növekvő vagy csökkenő sorrendbe rendezve adja vissza adatokat
- **ASC** növekvő (alapértelmezett)
- **DESC** csökkenő

Növekvő sorrend ASC

Jelenítsük meg az autók minden adatát a gyártók szerinti **növekvő** sorrendben.

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto` ASC;
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| ABC-123 | Ford | Focus | M1 | diesel | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| XXX-111 | Opel | Adam | M1 | benzin | piros |

Csökkenő sorrend DESC

Jelenítsük meg az autók minden adatát a gyártók szerinti **csökkenő** sorrendben.

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto` DESC;
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| ABC-123 | Ford | Focus | M1 | diesel | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |

Azonosak esetén...

A példán látható, hogy a Ford Focus és Ford Fiesta növekvő és csökkenő rendezés esetén is ugyanabban a sorrendben jelentek meg. Vesszővel felsorolhatunk több rendezési szempontot is.

Összetett rendezés példa 1

Jelenítse meg az autók adatát **gyártókszerint csökkenő**, míg **típus szerint növekvő** sorrendben!

MySQL

```
SELECT * FROM `autok`  
ORDER BY `gyarto` DESC, `tipus` ASC;
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|------------|--------|--------|-----------|-----------|-------|
| XXX-111 | Opel | Adam | M1 | benzin | piros |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |
| AA-AX-1234 | Ford | Fiesta | M1 | benzin | sárga |
| ABC-123 | Ford | Focus | M1 | diesel | kék |

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------------|-------------|-------------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | <i>NULL</i> | <i>NULL</i> | <i>NULL</i> |

Álnév probléma!

Jelenítsük meg a termékek nevét és a bruttó árat a bruttó szerinti növekvő sorrendben.

```
SELECT `nev`, `netto` * `afa` AS 'brutto'  
FROM `termekek`  
ORDER BY 'brutto' ASC;
```

logikai hiba

- A kód le fog futni, de nem a várt eredménnyel.

Álnév hiba eredménye

| `nev` | `brutto` | 'brutto' |
|--------------|--------------------|----------|
| 4K TV | 94.80999881029129 | brutto |
| Mobil 32GB | 56.80999928712845 | brutto |
| Mobil 128GB | 129.00999838113785 | brutto |
| Olcsó laptop | 51.10999935865402 | brutto |
| Drága laptop | 328.50999587774277 | brutto |
| Könyv | - | brutto |

- Az eredmény rendezett, de egy harmadik, mesterségesen generált mező alapján.
- Fontos, hogy az álnév backtick legyen, itt ennek hiányában nem működött a rendezés.

Rendezés számított mező alapján

MySQL

```
SELECT `nev`, `netto` * (1 + `afa`) AS `brutto`  
FROM `termekek`  
ORDER BY `brutto` ASC;
```

vagy

MySQL

```
SELECT `nev`, `netto` * (1 + `afa`)  
FROM `termekek`  
ORDER BY `netto` * (1 + `afa`) ASC;
```

Rendezés sorszám alapján

MySQL

```
SELECT `nev`, `netto`  
FROM `termekek`  
ORDER BY 2;
```

- A SELECT után felsorolt n-edik mező szerint is rendezhetőek az adatok
- A mezőket 1-től indexeli
- A fenti lekérdezés a második mező, azaz a nettó ár alapján rendez

Figyelem!

Nem az eredeti tábla, hanem a SELECT után felsorolt mezők sorszáma az, ami számít!

Rendezés és feltétel

Jelenítsük meg a **kék** színű autók minden adatát a gyártók szerinti **növekvő** sorrendben.

MySQL

```
SELECT * FROM `autok`  
WHERE `szin` = 'kék'  
ORDER BY `gyarto` ASC;
```

| rendszám | gyarto | tipus | kategoria | uzemanyag | szin |
|----------|--------|-------|-----------|-----------|------|
| ABC-123 | Ford | Focus | M1 | diesel | kék |
| AAA-555 | Honda | Jazz | M1 | hibrid | kék |

Rendezés és feltétel

Jelenítsük meg a **300 eurónál drágább** termékek nevét és **nettó árát**, utóbbi **szerint növekvő** sorrendben.

MySQL

```
SELECT `nev`, `netto`  
FROM `termekek`  
WHERE `netto` > 300  
ORDER BY `netto` ASC;
```

| nev | netto |
|--------------|-------|
| 4K TV | 499 |
| Mobil 128GB | 679 |
| Drága laptop | 1729 |

Tartalom I

7 Sorok számának limitálása (LIMIT)

Sorok számának limitálása (LIMIT)

A LIMIT segítségével megadhatjuk, hogy az eredmény hány sorát szeretnénk megkapni.

```
SELECT * FROM `tanulok`  
LIMIT 2;
```

MySQL

https://www.w3schools.com/php/php_mysql_select_limit.asp

Lapozó

Első N rekord kihagyása:

MySQL

```
SELECT id, vnev, knev  
FROM `tanulok`  
LIMIT 10, 5;
```

- Kihagyja az első 10 tanulót és onnantól kezdve jelenít meg legfeljebb 5 tanulót.
- Felhasználás a gyakorlatban:
 - Nagy mennyiségű adat esetén egy weboldalon nem jó ötlet az összes adatot egyszerre megjeleníteni, célszerű több oldalra bontani.
 - Amennyiben 1 oldalon 5 adatot jelenítünk meg, úgy a fenti példa a 3. oldal adatait kéri le
 - Az első két oldalon megjelenő 5-5, azaz összesen 10 rekord kerül kihagyásra, majd jön az aktuális oldal tartalma

Legjobb 3 tanuló

Listázzuk ki a három legjobb tanulót.

MySQL

```
SELECT id, vnev, knev  
FROM `tanulok`  
ORDER BY `atlag` DESC  
LIMIT 3;
```

- A tanulókat átlag alapján rendezi **csökkenő** sorrendbe
- Az átlag nem kerül megjelenítésre
- Az **ORDER BY** és **LIMIT** kombinálásával létrehozható toplista

A legmagasabb tanuló

Hogy hívják a legmagasabb tanulót?

MySQL

```
SELECT vnev, knev  
FROM `tanulok`  
ORDER BY `magassag` DESC  
LIMIT 1;
```

- A tanulókat a magasságuk alapján rendezi csökkenő sorrendbe
- A magasság nem kerül megjelenítésre
- Az **ORDER BY** és **LIMIT** kombinálásával meghatározható hogyan hívják a legmagasabb tanulót

A legidősebb tanuló

Hogy hívják a legidősebb tanulót?

MySQL

```
SELECT vnev, knev  
FROM `tanulok`  
ORDER BY `szuletesi_datum` ASC  
LIMIT 1;
```

- A tanulókat a születési dátumok alapján rendezi **növekvő** sorrendbe
 - Minél korábban született valaki, annál idősebb lesz
- Az **ORDER BY** és **LIMIT** kombinálásával meghatározható hogyan hívják legidősebb tanulót

Tartalom I

8 Számított mezők

Számított mezők

- SQL lekérdezésben lehetőség van számítások elvégzésére
- Szerepelhet benne:
 - Konstans érték: 1, 'hello', '2000-01-01', true
 - Egy mező az adatbázisból: `ar`
 - Valamilyen függvény: sqrt(9), round(1.975,2)

A DUAL "tábla"

MySQL

```
SELECT 10 + 5 AS `eredmeny`  
FROM dual;
```

- Előfordulhatnak olyan lekérdezések, amit nem táblától szeretnénk lekérdezni.
- A **dual** egy speciális „tábla”, ahonnan bármit lekérdezhetünk.
 - **Itt a backtick nem használható!**

MySQL

```
SELECT 10 + 5 AS `eredmeny`;
```

- Más adatbázisoknál kötelező
- A MySQL-ben elhagyható

Linkek:

- [SELECT - MySQL dokumentáció](#)

Aritmetikai operátorok

| Operátor | Művelet |
|------------|----------------------------------|
| - | Negatív előjel |
| * | Szorzás |
| / | (Valós) Osztás |
| MOD vagy % | Modulo operátor / Maradék képzés |
| DIV | Egész osztás |
| + | Összeadás |
| - | Kivonás |

- A műveletek precedencia (műveleti sorrend) szerinti sorrendben láthatóak

Linkek:

- Aritmetikai műveletek - MySQL dokumentáció

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------------|-------------|-------------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | <i>NULL</i> | <i>NULL</i> | <i>NULL</i> |

Számított mezők

Jelenítsük meg a termékek bruttó árait.

MySQL

```
SELECT
    `nev` AS `termek_nev`,
    `netto` * (1 + afa) AS `brutto`
FROM
    `termekek`;
```

- A lekérdezések során a tábla mezői felhasználhatóak különböző számításokhoz.

Szűrés számított mező alapján

Jelenítsük meg azokat a termékeket, melyek **bruttó ára** több, mint 400 euro

```
SELECT
    `nev` AS `termek_nev`,
    `netto` * (1 + afa) AS `brutto`
FROM `termekek`
WHERE `brutto` > 400;
```

Szintaktikai hiba

#1054 - A(z) 'brutto' oszlop ervenytelen 'where clause'-ben

Hiba!

- Az ANSI SQL szabvány szerint a **WHERE** záradékban nem használható ALIAS

Számított mezők feltételként

Jelenítsük meg azokat a termékeket, melyek **bruttó ára** több, mint 400 euro

MySQL

```
SELECT
    `nev` AS `termek_nev`,
    `netto` * (1 + afa) AS `brutto`
FROM `termekek`
WHERE `netto` * (1 + afa) > 400;
```

| `termek_nev` | `brutto` |
|--------------|----------|
| 4K TV | 633.73 |
| Mobil 128GB | 862.33 |
| Drága laptop | 2195.83 |

- A **WHERE** záradékban alkalmazhatóak számított értékek

Rendezés számított mező alapján

Jelenítsük meg a termékek nevét és a bruttó árat a bruttó szerinti növekvő sorrendben.

```
SELECT `nev`, `netto` * (1 + `afa`) AS 'brutto'  
FROM `termekek`  
ORDER BY 'brutto' ASC;
```

logikai hiba

- A kód le fog futni, de nem a várt eredménnyel.

Rendezés számított mező alapján

| `nev` | `brutto` | 'brutto' |
|--------------|--------------------|----------|
| 4K TV | 94.80999881029129 | brutto |
| Mobil 32GB | 56.80999928712845 | brutto |
| Mobil 128GB | 129.00999838113785 | brutto |
| Olcsó laptop | 51.10999935865402 | brutto |
| Drága laptop | 328.50999587774277 | brutto |
| Könyv | - | brutto |

- Az eredmény rendezett, de egy harmadik, mesterségesen generált mező alapján.
- Fontos, hogy az álnév backtick legyen, itt ennek hiányában nem működött a rendezés.

Rendezés számított mező alapján

Jelenítsük meg a termékek nevét és a bruttó árat a bruttó szerinti növekvő sorrendben.

MySQL

```
SELECT `nev`, `netto` * (1 + `afa`) AS `brutto`  
FROM `termekek`  
ORDER BY `netto` * (1 + `afa`) ASC;
```

- A rendezési feltétel kiszámítása elvégezhető az **ORDER BY** záradékban

MySQL

```
SELECT `nev`, `netto` * `afa` AS `brutto`  
FROM `termekek`  
ORDER BY `brutto` ASC;
```

- Az **ORDER BY** záradékban használható a **SELECT**-ben meghatározott álnév

Rendezés számított mező alapján

| <code>`nev`</code> | <code>`brutto`</code> |
|--------------------|-----------------------|
| Könyv | <i>NULL</i> |
| Olcsó laptop | 51.10999935865402 |
| Mobil 32GB | 56.80999928712845 |
| 4K TV | 94.80999881029129 |
| Mobil 128GB | 129.00999838113785 |
| Drága laptop | 328.50999587774277 |

- Rendezéskor a NULL értékeket mindennél kisebbnek tekinti a MySQL
- Növekvő sorrend esetében az elsők között szerepel
- Csökkenő sorrend esetében az utolsók között szerepel

Tartalom I

- 9 Matematikai függvények
 - Egyszerű matematikai függvények
 - Kerekítés

Tartalom

- 9 Matematikai függvények
 - Egyszerű matematikai függvények
 - Kerekítés

Matematikai függvények

`ABS(x)` $|x|$ abszolút érték

`MOD(x,y)` maradékos osztás

`POW(x,y)` x^y hatványozás

`POWER(x,y)` x^y hatványozás

`SQRT(x)` \sqrt{x} gyök

<https://dev.mysql.com/doc/refman/8.0/en/mathematical-functions.html>

Függvényhasználat: SQRT()

A lekérdezésben használhatunk függvényeket, például a gyök függvényt!

MySQL

```
SELECT SQRT(9) AS `negyzetgyok`  
FROM DUAL;
```

| negyzetgyok |
|-------------|
| 3 |

Linkek:

- [MySQL dokumentáció: Matematikai függvények](#)

PI()

PI()

- Megadja a π (pi) értékét.
- Alapértelmezetten 7 számjegyet jelenít meg
 - ebből 1 számjegy az egész résznek,
 - és 6 számjegy a tört résznek.
- Ennél nagyobb pontossággal tárolja és számol vele.

```
SELECT PI() as `pite` FROM DUAL;
```

MySQL

pite

3.141593

Tartalom

- 9 Matematikai függvények
 - Egyszerű matematikai függvények
 - Kerekítés

Kerekítő függvények

CEILING(x) $\lceil x \rceil$ felső egész rész

CEIL(x) alias a CEILING() függvényre

FLOOR(x) $\lfloor x \rfloor$ alsó egész rész

ROUND(x,n) matematikai kerekítés

TRUNCATE(x,n) nem kerekít, levágja a tizedes jegyeket

Linkek:

- MySQL dokumentáció: Matematikai függvények

Kerekítés ROUND(x,d)

x Kerekítendő érték

d tizedesek száma

A bruttó árat két tizedesre kerekítve jelenítse meg!

MySQL

```
SELECT ROUND(`netto` * (1 + afa) ,2) AS `brutto`  
FROM `termekek`;
```

- Alapvetően a matematikai kerekítést alkalmazza, 5-től felfelé kerekít
- Lebegőpontos számábrázolás esetén bizonyos rendszereken előfordul, hogy a "Round to Even", más néven "Banker's Rounding" módszert alkalmazhatja

Linkek:

- MySQL dokumentáció: Az ROUND() függvény
- Wikipedia: Szimmetrikus kerekítés (Banker's Rounding)

ROUND() példák

MySQL

```
SELECT ROUND(123.4567) as `eredmeny`  
FROM DUAL;
```

eredmeny

123

- Ha a második paraméter 0, vagy nincs, akkor egészre kerekít

MySQL

```
SELECT ROUND(123.4567,1) as `eredmeny`  
FROM DUAL;
```

eredmeny

123.5

- A második paraméter 1, így egy tizedesre kerekít

MySQL

```
SELECT ROUND(123.4567,-1) as `eredmeny`  
FROM DUAL;
```

eredmeny

120

- Mivel a második paraméter -1, így a tizes helyiértékű számra kerekíti

Linkek:

- [MySQL dokumentáció: Az ROUND\(\) függvény](#)

CEIL(), FLOOR(), és ROUND() összehasonlítása

MySQL

```
SELECT CEIL(222.111) as `eredmeny`  
FROM DUAL;
```

eredmeny

223

- A CEIL() függvény visszaadja a **tőle nem kisebb** legkisebb egész számot

MySQL

```
SELECT FLOOR(111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

111

- A FLOOR() függvény visszaadja a **tőle nem nagyobb** legnagyobb egész számot

MySQL

```
SELECT ROUND(111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

112

- A ROUND() függvény kerekítést alkalmaz

Linkek:

- [MySQL dokumentáció: Az ROUND\(\) függvény](#)

CEIL(), FLOOR(), és ROUND() negatív számokkal

MySQL

```
SELECT CEIL(-222.111) as `eredmeny`  
FROM DUAL;
```

eredmeny

-222

- A CEIL() függvény visszaadja a **tőle nem kisebb** legkisebb egész számot

MySQL

```
SELECT FLOOR(-111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

-112

- A FLOOR() függvény visszaadja a **tőle nem nagyobb** legnagyobb egész számot

MySQL

```
SELECT ROUND(-111.888) as `eredmeny`  
FROM DUAL;
```

eredmeny

-112

- A ROUND() függvény kerekítést alkalmaz

Linkek:

Tartalom I

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

CONCAT()

```
CONCAT(str1,str2,...)
```

- Összefűzi az argumentumként kapott értékeket
- A számokat átalakítja szöveggé
- Amennyiben tartalmaz **NULL** értéket, úgy a végeredmény is **NULL** lesz

Linkek:

- MySQL dokumentáció: [CONCAT\(\)](#)

CONCAT() példák

MySQL

```
SELECT CONCAT('A', 'B', 'C', 'D') as `e`  
FROM DUAL;
```

e

ABCD

- Több, mint két argumentum is megadható

MySQL

```
SELECT CONCAT('Hello', ' ', 'World') as `e`  
FROM DUAL;
```

e

Hello World

- A szóköz külön argumentumként lett megadva

MySQL

```
SELECT CONCAT(15, ' cm') as `e`  
FROM DUAL;
```

e

15 cm

- A szóköz a ' cm' értékben található meg

CONCAT() példák

MySQL

```
SELECT
  CONCAT(`magassag` / 100, ' m') as `magassag_meterben`
FROM `tanulok`;
```

| magassag_meterben |
|-------------------|
| 1,72 m |
| 1,83 m |
| 1,85 m |
| ... |

- A `magassag` a `tanulok` tábla egyik oszlopa
- A magasság cm-ből m-re át lett számítva
- A szóköz a ' m' értékben található meg

CONCAT() és rendezés

logikai hiba

```
SELECT
    `nev`,
    CONCAT(ROUND(`netto` * (1 + afa)), ' EUR') AS `eur`
FROM `termekek`
WHERE `netto` IS NOT NULL
ORDER BY `eur` ASC;
```

| | |
|--------------|----------|
| Drága laptop | 2058 EUR |
| Olcsó laptop | 320 EUR |
| Mobil 32GB | 356 EUR |
| 4K TV | 594 EUR |
| Mobil 128GB | 808 EUR |

- A CONCAT miatt a rendezés szövegek alapján történik
- Így akár '2' > '1 000 000' igaz (szöveges összehasonlítás)
- A sorrend hibás lesz

CONCAT() és rendezés

MySQL

```
SELECT
    `nev`,
    CONCAT(ROUND(`netto` * (1 + afa)), ' EUR') AS `eur`
FROM `termekek`
WHERE `netto` IS NOT NULL
ORDER BY ROUND(`netto` * (1 + afa)) ASC;
```

| | |
|--------------|----------|
| Olcsó laptop | 320 EUR |
| Mobil 32GB | 356 EUR |
| 4K TV | 594 EUR |
| Mobil 128GB | 808 EUR |
| Drága laptop | 2058 EUR |

- Szabvány szerint használhatnánk az alias a rendezésben
- A CONCAT-ben található rész szerint kell rendezni:

ROUND(`netto` * (1 + afa))

CONCAT_WS()

```
CONCAT_WS(separator, str1, str2, ...)
```

- "Concatenate With Separator"
- Összefűzi az argumentumként kapott értékeket
- Az első argumentum az elválasztó karakter
- Az elválasztó karaktert a legvégére nem teszi ki

```
SELECT CONCAT_WS('*', 'alma', 'barack', 'eper') AS `e`  
FROM DUAL;
```

MySQL

e

alma*barack*eper

Linkek:

- [MySQL dokumentáció: CONCAT_WS\(\)](#)

Logikai konstansok

- A TRUE logikai konstans értéke: 1
- A False logikai konstans értéke: 0
- A kis- és nagybetűkre nem érzékeny

MySQL

```
SELECT  
    TRUE, true, True, TrUe, FALSE, false, False, fALSe  
FROM DUAL;
```

| TRUE | true | True | TrUe | FALSE | false | False | fALSe |
|------|------|------|------|-------|-------|-------|-------|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

Linkek:

- [MySQL dokumentáció: Boolean Literals](#)

Szövegek és számok összehasonlítása

MySQL

```
SELECT 2 > 1999999 AS `eredmeny`  
FROM DUAL;
```

0 - FALSE

- Számként összehasonlítva az 1999999 a nagyobb

MySQL

```
SELECT '2' > '1999999' AS `eredmeny`  
FROM DUAL;
```

1 - TRUE

- Szöveggént összehasonlítva a 2 a nagyobb
- Az első karaktert összehasonlítva eldöntötte, hogy '2' > '1'

MySQL

```
SELECT '2' > '2000000' AS `eredmeny`  
FROM DUAL;
```

0 - FALSE

- Az első karaktert összehasonlítva nem állapítható meg a nagyobb, mert '2' = '1'
- Az lesz a nagyobb, amiben még találhatóak további karakterek

Összefűtés és rendezés

logikai hiba

```
SELECT
  `nev`,
  CONCAT(ROUND(`netto`*(1+`afa`),2), `penznem`) AS `brutto`
FROM `termekek`
ORDER BY `brutto` ASC;
```

| nev | brutto |
|--------------|------------|
| Könyv | NULL |
| Drága laptop | 2057.51EUR |
| Olcsó laptop | 320.11EUR |
| Mobil 32GB | 355.81EUR |
| 4K TV | 593.81EUR |
| Mobil 128GB | 808.01EUR |

- A CONCAT miatt a brutto oszlop értékeit szöveggént hasonlítja össze
- A sorrend nem megfelelő

Összefűtés és rendezés

MySQL

```
SELECT
    `nev`,
    CONCAT(ROUND(`netto`*(1+`afa`),2), `penznem`) AS `brutto`
FROM `termekek`
ORDER BY ROUND(`netto`*(1+`afa`),2) ASC;
```

| nev | brutto |
|--------------|------------|
| Könyv | NULL |
| Olcsó laptop | 320.11EUR |
| Mobil 32GB | 355.81EUR |
| 4K TV | 593.81EUR |
| Mobil 128GB | 808.01EUR |
| Drága laptop | 2057.51EUR |

- A kerekített értéket látjuk, így célszerű annak megfelelően rendezni
- A CONCAT ugyan szerepel a SELECT-ben, de az ORDER BY záradékban már nem
- A sorrend így már helyes

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

UPPER()

```
UPPER(str)
```

- Nagybetűssé alakítja a szöveget (str)
- Alapértelmezetten a latin1 kódolást (cp1252 West European) használja
- Amennyiben a tábla karakterkódolás jól van megadva multibyte karaktereket is jól kezel

MySQL

```
SELECT UPPER('heLLo') FROM dual;
```

e

HELLO

LOWER()

LOWER(str)

- Kisbetűssé alakítja a szöveget (str)
- Alapértelmezetten a latin1 kódolást (cp1252 West European) használja
- Amennyiben a tábla karakterkódolás jól van megadva multibyte karaktereket is jól kezel

MySQL

```
SELECT LOWER('heLLo') FROM dual;
```

e

hello

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - **Hossz**
 - Hossz
 - Keresés
 - Csere
 - Kivágás

LENGTH()

LENGTH(str)

- Megadja a szöveg (str), hosszát **byteokban**
- A multibyte karaktereket **többször** számolja

LENGTH() példák

MySQL

```
SELECT LENGTH('car') as `e`  
FROM DUAL;
```

e

3

- Az egy byteos karakterek (ASCII első 128 karaktere) hossza megegyezik a karaktereinek számával

logikai hiba

```
SELECT LENGTH('autó') as `e`  
FROM DUAL;
```

e

5

- Az "autó" 4 betűs szó, de a hosszú "ó" multibytos karakter, így lesz a végeredmény 5

CHAR_LENGTH()

CHAR_LENGTH(str)

- Megadja a szöveg (str), hosszát **karakterekben**
- A multibyte karaktereket **egyszer** számolja
- Szinonímák erre a függvényre:
 - **CHARACTER_LENGTH**(str)

CHAR_LENGTH() példák

```
SELECT CHAR_LENGTH('car') as `e`  
FROM DUAL;
```

MySQL

e

3

- Az egy byteos karakterek (ASCII első 128 karaktere) hossza megegyezik a karaktereinek számával

```
SELECT CHAR_LENGTH('autó') as `e`  
FROM DUAL;
```

MySQL

e

4

- Az "autó" 4 betűs szó, amit helyesen megállapított a függvény

LENGTH() és CHAR_LENGTH() összehasonlítása

```
SELECT LENGTH('árvíztűrőtükörfúrógép') as `e`  
FROM DUAL;
```

logikai hiba

e

30

```
SELECT CHAR_LENGTH('árvíztűrőtükörfúrógép') as `e`  
FROM DUAL;
```

MySQL

e

21

- A **LENGTH()** a byteok számát, míg a **CHAR_LENGTH()** a karakterek számát adja meg, így utóbbi a multibyteos karakterek esetén is helyesen állapítja meg a szöveg hosszát.

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - **Hossz**
 - Keresés
 - Csere
 - Kivágás

FORMAT()

```
FORMAT(X,D[,locale])
```

- Az X számot formázza ezres csoportosítással
- A tizedesek számát a D határozza meg
- A nyelvi beállítás határozza meg,
 - hogy tizedes pontot ('en_US'), vagy
 - hogy tizedes vesszőt ('hu_HU') használjon

MySQL

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

LOCATE()

```
LOCATE(substr, str)
```

vagy

```
LOCATE(substr, str, pos)
```

- Megkeresi a keresett szöveg (substr), a kezdő pozícióját a szövegben (str) a megadott számú (pos) karaktertől kezdve
- Az indexelés 1-től kezdődik
- Ha nem találja meg 0-t ad vissza.
- Az eredeti szöveget nem módosítja
- Szinonímák erre a függvényre:
 - **POSITION**(substr **IN** str)

LOCATE() példák

```
SELECT LOCATE('vár', 'Székesfehérvár') as `e`  
FROM DUAL;
```

MySQL

e

12

- A "vár" szöveg "v" betűje a 12. karakter

```
SELECT LOCATE('Székesfehérvár', 'vár') as `e`  
FROM DUAL;
```

MySQL

e

0

- A "vár" szöveg nem tartalmazza a "Székesfehérvár" szöveget, így az eredmény 0

LOCATE() példák

MySQL

```
SELECT LOCATE('é', 'Székesfehérvár') as `e`  
FROM DUAL;
```

e

3

- Az "é" betű a 3. karakter a szó legelejétől keresve

MySQL

```
SELECT LOCATE('é', 'Székesfehérvár', 3) as `e`  
FROM DUAL;
```

e

3

- Az "é" betű a 3. karakter a szó 3. karakterétől keresve

MySQL

```
SELECT LOCATE('é', 'Székesfehérvár', 4) as `e`  
FROM DUAL;
```

e

10

- Az "é" betű a 10. karakter a szó 4. karakterétől keresve

Tartalom

10 Szöveg függvények

- Összefűzés
- Kis- és nagybetűk
- Hossz
- Hossz
- Keresés
- Csere
- Kivágás

REPLACE()

```
REPLACE(str,from_str,to_str)
```

- Lecseréli a szövegben (str), az összes előfordulását a keresett szövegrésznek (from_str) az új szövegre (to_str)
- Az eredeti szöveget nem módosítja

REPLACE() példák

MySQL

```
SELECT  
  REPLACE('Székesfehérvár', 'é', 'e') as `e`  
FROM DUAL;
```

e

Szekesfeheervár

- Az "é" betű lett lecserélve az "e" betűre

MySQL

```
SELECT  
  REPLACE(REPLACE('Székesfehérvár', 'é', 'e'), 'á', 'a') as `e`  
FROM DUAL;
```

e

Szekesfehervar

- A függvény többszöri egymásba ágyazásával több karakter is lecserélhető

Tartalom

- 10 Szöveg függvények
 - Összefűzés
 - Kis- és nagybetűk
 - Hossz
 - Hossz
 - Keresés
 - Csere
 - Kivágás

SUBSTRING()

```
SUBSTRING(str,pos,len)
```

- Kivág egy részt a szövegből (str), a kezdő pozíciótól (pos) kezdve megadott számú (len) karaktert
- Az eredeti szöveget nem módosítja
- Szinonímák erre a függvényre:
 - SUBSTR()
 - MID()

SUBSTRING() példák

```
SELECT SUBSTRING('Székesfehérvár',7) as `e`  
FROM DUAL;
```

MySQL

e

fehérvár

- A 1en elhagyásával a szöveget a pos-tól a legvégéig veszi

```
SELECT SUBSTRING('Székesfehérvár',7,5) as `e`  
FROM DUAL;
```

MySQL

e

fehér

- A 7. karaktertől vesz 5 karaktert

```
SELECT SUBSTRING('Székesfehérvár',-3,3) as `e`  
FROM DUAL;
```

MySQL

e

vár

- Negatív pos esetén hátulról lép vissza, majd a 1en-ben meghatározott karaktert veszi, annak elhagyásával a végéig

Tartalom I

- 11 Dátum és idő függvények
 - Dátum/idő részének kinyerése
 - Aktuális dátum/idő

Tartalom

- 11 Dátum és idő függvények
 - Dátum/idő részének kinyerése
 - Aktuális dátum/idő

Dátum/Idő részének kinyerése

- `DATE()` Megadja a dátum részt egy dátum/időből
- `TIME()` Megadja az idő rész egy dátum/időből
- `YEAR()` Megadja az évet
- `MONTH()` Megadja a hónapot
- `DAY()` Megadja a napot
- `HOUR()` Megadja az órát
- `MINUTE()` Megadja a percet
- `SECOND()` Megadja a másodpercet
- `WEEKDAY()` Megadja, hogy az adott dátum a hét hányadik napja
 - Paramétere lehet egy mező `YEAR(`született`)`,
 - vagy konkrét érték `YEAR('2022-01-12')`

Linkek:

- MySQL dokumentáció: Dátum és idő függvények

A YEAR() függvény használata

Melyik évben született az 1-es azonosítójú tanuló?

MySQL

```
SELECT YEAR(`szul_ido`)  
FROM `tanulok`  
WHERE `id` = 1;
```

- A `szul_ido` a születési dátumokat tartalmazza (pl.: '2003-03-16')
- A `YEAR()` függvény az évet nyeri ki belőle. (pl.: 2003)

A YEAR() függvény használata

A hét melyik napján született az 1-es azonosítójú tanuló?

MySQL

```
SELECT WEEKDAY(`szul_ido`)  
FROM `tanulok`  
WHERE `id` = 1;
```

- A `szul_ido` a születési dátumokat tartalmazza (pl.: '2003-03-16')
- A `WEEKDAY()` függvény a nap sorszámát határozza meg. (pl.: 6)
 - 0 - Hétfő
 - 1 - Kedd
 - ...
 - 6 - vasárnap

Tartalom

- 11 Dátum és idő függvények
 - Dátum/idő részének kinyerése
 - Aktuális dátum/idő

Aktuális dátum és idő

- Aktuális dátum
 - `CURDATE()`
 - `CURRENT_DATE()`
- Aktuális idő
 - `CURTIME()`
 - `CURRENT_TIME()`
- Aktuális dátum és idő
 - `NOW()`

Linkek:

- [MySQL dokumentáció: Dátum és idő függvények](#)

Tartalom I

12 Egyéb hasznos függvények

COALESCE()

```
COALESCE(value,...)
```

- Visszaadja az első **nem NULL** értéket

```
SELECT COALESCE(NULL, NULL, NULL, 'A', 'B') AS `e`  
FROM DUAL;
```

MySQL

e

A

Tartalom I

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

Összesítő függvények

Az összesítő (aggregált) függvények a meghatározott kifejezésen hajtanak végre különböző műveleteket.

- Alapértelmezetten a NULL értékeket nem veszik számításba
- Gyakran használt összesítő függvények:
 - COUNT()
 - SUM()
 - AVG()
 - MIN()
 - MAX()

Linkek:

- MySQL dokumentáció: Összesítő függvények

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

A COUNT() függvény

Megszámolja a lekérdezés által visszaadott sorokban a nem NULL értékeket.

- Van lehetőség NULL beleszámítására is
- Amennyiben nincs a feltételeknek megfelelő találat, úgy 0 lesz a függvény kimenete
- A MySQL nem csak a számokat tartalmazó mezőkön értelmezi
- Az eredmény BIGINT típusú lesz

Linkek:

- MySQL dokumentáció: [COUNT\(\)](#)

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------|---------|------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | NULL | NULL | NULL |

Példa: COUNT(`netto`) példa

MySQL

```
SELECT COUNT(`netto`) AS `darab`  
FROM `termekek`;
```

darab

5

- Ahol a netto értéke **NULL**, azt a sort kihagyja a számításból.

Példa: COUNT() - szöveget tartalmazó oszlopon

MySQL

```
SELECT COUNT(`penznem`) AS `db`  
FROM `termekek`;
```

db

5

- Ahol a penznem értéke **NULL**, azt a sort kihagyja a számításból.

Példa: COUNT(*)

MySQL

```
SELECT COUNT(*) AS `db_csillag`  
FROM `termekek`;
```

| db_csillag |
|------------|
| 6 |

- A **COUNT(*)** a visszaadott sorok számát számolja meg, így a NULL értékeket is beleveszi a számításába!

Példa: COUNT() - a tábla elsődleges kulcsára alkalmazva

MySQL

```
SELECT COUNT(`id`) AS `darab_id_szerint`  
FROM `termekek`;
```

| darab_id_szerint |
|------------------|
| 6 |

- Az elsődleges kulcs sosem lehet **NULL**
- Érdemes az elsődleges kulcsot megadni paraméterként
- Az elsődleges kulcs gyakran id vagy Azon néven szerepel

Egyedi értékek az összesítő függvényekben

Hány **különböző** kategória található a táblában?

MySQL

```
SELECT COUNT(DISTINCT `kategoria`) AS `db_kategoria`  
FROM `termekek`;
```

| db_kategoria |
|--------------|
| 4 |

- Amennyiben a DISTINCT kulcsszót a COUNT() függvényen belül helyezzük el, úgy az azonos értékeket egyszer veszi csak számításba.

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

A SUM() függvény

Összeadja a meghatározott kifejezés értékeit.

- A **DISTINCT** megadásával csak az egyedi értékeket összegzi
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek összege is **NULL** lesz

Linkek:

- MySQL dokumentáció: SUM()

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------------|-------------|-------------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | <i>NULL</i> | <i>NULL</i> | <i>NULL</i> |

Példa: SUM()

Mennyi a termékek *nettó értéke* **összesen**?

MySQL

```
SELECT SUM(`netto`) AS `ossz`  
FROM `termekek`;
```

OSSZ

3475

- A könyv netto értéke **NULL**
 - Nem adta hozzá az eredményhez
 - Nem lett a végeredmény **NULL**

Példa: SUM() - NULL értékek kihagyásával

MySQL

```
SELECT SUM(`netto`) AS `ossz`  
FROM `termekek`  
WHERE `netto` IS NOT NULL;
```

OSSZ

3475

- A könyv sora kimarad a számításból

Példa: SUM() - NULL értékekkel

MySQL

```
SELECT SUM(`netto`) AS `ossz`  
FROM `termekek`  
WHERE `netto` IS NULL;
```

OSSZ

NULL

- A könyv sorában lesz egyedül **NULL** érték, a végeredmény is **NULL** lett

Példa: SUM() - feltétellel

Mennyibe kerülnek a mobilok?

MySQL

```
SELECT SUM(`netto`) AS `ossz_mobil`  
FROM `termekek`  
WHERE `kategoria` = 'mobil';
```

ossz_mobil

978

- Az összegzés előtt szűr a **WHERE** feltétel alapján
- Csak azokat a sorokat veszi, ahol a **kategoria** értéke "mobil"

Példa: SUM() - számított mező összegzése

Mennyi a termékek **bruttó** értéke?

MySQL

```
SELECT  
    ROUND( SUM(`netto` * (1 + `afa`)) , 2) AS `ossz`  
FROM  
    `termekek`;
```

OSSZ

4135.25

- Kiszámítja a bruttó értéket: ``netto` * (1 + `afa`)`
- A számított értékeket összegzi: `SUM()`
- A végeredményt kerekíti 2 tizedesre: `ROUND()`

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- **AVG**
- MIN/MAX

Az AVG() függvény

Meghatározza a megadott kifejezés átlagát.

- A **DISTINCT** megadásával csak az egyedi értékeket átlagolja
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek átlaga is **NULL** lesz

Linkek:

- MySQL dokumentáció: [AVG\(\)](#)

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------|---------|------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | NULL | NULL | NULL |

Példa: AVG()

Mennyi a termékek *nettó árának* az **átlaga**?

MySQL

```
SELECT AVG(`netto`) AS `netto_atlag`  
FROM `termekek`;
```

| netto_atlag |
|-------------|
| 695 |

Példa: AVG()

Mennyi a termékek *bruttó árának* az **átlaga**?

MySQL

```
SELECT  
    AVG(`netto` * ( 1 + `afa` ) ) AS `brutto_atlag`  
FROM  
    `termekek`;
```

brutto_atlag

827.0499983429909

Tartalom

13 Aggregált (összesítő) függvények

- COUNT
- SUM
- AVG
- MIN/MAX

A MIN() függvény

A megadott kifejezés legkisebb értékével tér vissza.

- Szöveggel is működik, az eredmény a karakterkódolástól függhet
- A **DISTINCT** megadásával csak az egyedi értékeket veszi figyelembe, de itt nem számít, mivel csak egy értéket ad úgyis vissza
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek minimuma is **NULL** lesz

Linkek:

- MySQL dokumentáció: MIN()

A MAX() függvény

A megadott kifejezés legnagyobb értékével tér vissza.

- Szöveggel is működik, az eredmény a karakterkódolástól függhet
- A **DISTINCT** megadásával csak az egyedi értékeket veszi figyelembe, de itt nem számít, mivel csak egy értéket ad úgyis vissza
- Amennyiben a lekérdezés egyetlen sorral sem tér vissza, úgy **NULL** lesz az eredmény
- A **NULL** értékek maximum is **NULL** lesz

Linkek:

- MySQL dokumentáció: MAX()

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------|---------|------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | NULL | NULL | NULL |

Példa: MIN()

Mennyi a **legolcsóbb** termék *nettó ára*?

MySQL

```
SELECT MIN(`netto`) AS `min_netto`  
FROM `termekek`;
```

| min_netto |
|-----------|
| 269 |

Példa: MIN()

Mi a *neve* annak a terméknek, ami **ABC**-ben az **első**?

```
SELECT MIN(`nev`) AS `min_nev`  
FROM `termekek`;
```

MySQL

min_nev

4K TV

Példa: a MIN() hibás használata

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

```
SELECT nev AS `min_nev`, MIN(`netto`)
FROM `termekek`;
```

logikai hiba

#1140 - In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'pelda.termek.nev'; this is incompatible with sql_mode=only_full_group_by

Hiba!

- Senki sem kérte az árat, hanem csak a nevet

Példa: a MIN() hibás használata

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

```
SELECT nev  
FROM `termekek`  
WHERE MIN(`netto`);
```

logikai hiba

```
#1111 - Invalid use of group function
```

Hiba!

- Az összesítő függvények nem használhatóak a **WHERE** záradékban

Példa: a MIN() helyett ORDER BY és LIMIT

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

```
SELECT nev AS `min_nev`  
FROM `termekek`  
ORDER BY `netto`  
LIMIT 1;
```

logikai hiba

min_nev

Könyv

- A "legolcsóbb" nettó érték a **NULL** lesz a rendezés szerint
- Előre ki kell szűrni a **NULL** értékeket

Példa: a MIN() helyett ORDER BY és LIMIT

Mi a **neve** a *nettó ár* szerint **legolcsóbb** terméknek?

MySQL

```
SELECT nev AS `min_nev`  
FROM `termekek`  
WHERE `netto` IS NOT NULL  
ORDER BY `netto`  
LIMIT 1;
```

min_nev

Olcsó laptop

Példa: MIN() és MAX() egy lekérdezésben

MySQL

```
SELECT
  MIN(`netto`) AS `min_ar`,
  MAX(`netto`) AS `max_ar`
FROM
  `termekek`;
```

| min_ar | max_ar |
|--------|--------|
| 269 | 1729 |

- Egy lekérdezésben több összesítő függvény is szerepelhet
- Nem csak a **MIN()** és a **MAX()**

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------------|-------------|-------------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | <i>NULL</i> | <i>NULL</i> | <i>NULL</i> |

Tartalom I

14 Csoportosítás (GROUP BY)

Csoportosítás

- A GROUP BY
- <https://dev.mysql.com/doc/refman/8.0/en/group-by-functions.html>

Figyelem!

Csoportosításnál csak az szerepelhet a SELECT után, ami vagy szerepel a GROUP BY után, vagy összesítő függvényben van.

GROUP BY példa 1.

Melyik kategóriában hány termék található meg?

vagy

Határozza meg kategóriánként a termékek számát!

MySQL

```
SELECT COUNT(`id`) AS `db`  
FROM `termekek`  
GROUP BY `kategoria`;
```

- Illene megadni a kategóriát is!

| db |
|----|
| 1 |
| 2 |
| 2 |
| 1 |

GROUP BY példa 1.

Melyik kategóriában hány termék található meg?

vagy

Határozza meg kategóriánként a termékek számát!

MySQL

```
SELECT `kategoria`, COUNT(`id`) AS `db`  
FROM `termekek`  
GROUP BY `kategoria`;
```

| kategoria | db |
|-----------|----|
| tv | 1 |
| mobil | 2 |
| laptop | 2 |
| konyv | 1 |

GROUP BY példa 2.

Melyik kategóriában mennyibe kerülnek **átlagosan** a termékek?
vagy

Határozza meg kategóriánként az **átlagos** árat.

MySQL

```
SELECT  AVG(`netto`) AS `atlag`  
FROM    `termekek`  
GROUP BY `kategoria`;
```

- Itt a kategóriát is meg kell jeleníteni, különben nem tudnánk melyikhez tartozik.

GROUP BY példa 2.

Melyik kategóriában mennyibe kerülnek **átlagosan** a termékek?

vagy

Határozza meg kategóriánként az **átlagos** árat.

MySQL

```
SELECT `kategoria`, AVG(`netto`) AS `atlag`  
FROM `termekek`  
GROUP BY `kategoria`;
```

| kategoria | atlag |
|-----------|-------|
| tv | 499 |
| mobil | 489 |
| laptop | 999 |
| konyv | NULL |

GROUP BY példa 3.

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`;
```

logikai hiba

| nev | legolcsobb |
|-------|------------|
| 4K TV | 269 |

- **Gyakori hiba, hogy kimarad a GROUP BY!**
- Itt megkeresi a legolcsóbb árat és hozzá írja egy tetszőleges termék nevét.

GROUP BY példa 3. (sql_mode=only_full_group_by)

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`;
```

logikai hiba

#1140 - In aggregated query without GROUP BY, expression #1 of SELECT list contains nonaggregated column 'pelda.termekek.nev'; this is incompatible with sql_mode=only_full_group_by

Hiba!

- Összesítő lekérdezés csoportosítás nélkül lehetséges, de itt
- Az 1-es számú kifejezés a SELECT-ben (``nev``) nem összesítő függvényben szerepel

GROUP BY példa 4.

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legolcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`  
GROUP BY `penznem`;
```

logikai hiba

| nev | legolcsobb |
|-------|------------|
| Könyv | - |
| 4K TV | 269 |

- **Oda kell figyelni, hogy mi alapján csoportosítunk!**
- A pénznemben itt csak EUR és NULL szerepel.
- A kategóriát kell megjeleníteni a nev helyett!

GROUP BY példa 4. (sql_mode=only_full_group_by)

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legocslcsóbb** árat.

```
SELECT `nev`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`  
GROUP BY `penznem`;
```

logikai hiba

#1055 - Expression #1 of SELECT list is not in GROUP BY clause and contains nonaggregated column 'pelda.termek.nev' which is not functionally dependent on columns in GROUP BY clause; this is incompatible with sql_mode=only_full_group_by

Hiba!

- Az 1-es számú kifejezés a SELECT-ben (``nev``)
 - nem szerepel a GROUP BY záradékban
 - nem összesítő függvényben szerepel

GROUP BY példa 4.

Melyik kategóriában mennyibe kerül a **legolcsóbb** termék?

vagy

Határozza meg kategóriánként a **legocslcsóbb** árat.

MySQL

```
SELECT `kategoria`, MIN(`netto`) AS `legolcsobb`  
FROM `termekek`  
GROUP BY `kategoria`;
```

| kategoria | legolcsobb |
|-----------|------------|
| tv | 499 |
| mobil | 299 |
| laptop | 269 |
| könyv | - |

GROUP BY példa 5.

Melyik kategóriában mennyibe kerül a **legdrágább** termék?

vagy

Határozza meg kategóriánként az **legdrágább** árat.

MySQL

```
SELECT `kategoria`, MAX(`netto`) AS `legdragabb`  
FROM `termekek`  
GROUP BY `kategoria`;
```

| kategoria | legdragabb |
|-----------|------------|
| könyv | - |
| laptop | 1729 |
| mobil | 679 |
| tv | 499 |

GROUP BY példa 6.

**Melyik kategóriában mennyibe kerül a legdrágább termék?
Ahol nincs ár az ne jelenjen meg!**

MySQL

```
SELECT `kategoria`, MAX(`netto`) AS `legdragabb`  
FROM `termekek`  
WHERE `netto` IS NOT NULL  
GROUP BY `kategoria`;
```

| kategoria | legdragabb |
|-----------|------------|
| laptop | 1729 |
| mobil | 679 |
| tv | 499 |

GROUP BY példa 7.

Melyik kategóriában mennyibe kerül a **legdrágább** termék?
Ahol nincs ár az ne jelenjen meg! **A bruttó ár** jelenjen meg!

MySQL

```
SELECT
    `kategoria`,
    MAX(`netto` * (1 + `afa`) ) AS `max_brutto`
FROM `termekek`
WHERE `netto` IS NOT NULL`
GROUP BY `kategoria`;
```

| kategoria | max_brutto |
|-----------|------------|
| laptop | 1729 |
| mobil | 679 |
| tv | 499 |

GROUP BY példa 8.

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint növekvő** sorrendben legyenek rendezettek!

MySQL

```
SELECT
    `kategoria`,
    MAX(`netto` * (1 + `afa`) ) AS `max_brutto`
FROM `termekek`
WHERE `netto` IS NOT NULL
GROUP BY `kategoria`
ORDER BY MAX( `netto` * (1 + `afa`) ) ASC;
```

GROUP BY példa 8. eredménye

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint növekvő** sorrendben legyenek rendezettek!

| kategoria | max_brutto |
|-----------|--------------------|
| tv | 593.8099988102913 |
| mobil | 808.0099983811378 |
| laptop | 2057.5099958777428 |

GROUP BY példa 9.

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint csökkenő** sorrendben legyenek rendezettek!

MySQL

```
SELECT
    `kategoria`,
    MAX( `netto` * (1 + afa) ) AS `max_brutto`
FROM `termekek`
WHERE `netto` IS NOT NULL
GROUP BY `kategoria`
ORDER BY `max_brutto` DESC;
```

- MySQL-ben használhatjuk az álnevet rendezéshez.

GROUP BY példa 9. eredménye

Melyik kategóriában mennyibe kerül a **legdrágább** termék? Ahol nincs ár az ne jelenjen meg! A bruttó ár jelenjen meg! Az adatok **ár szerint csökkenő** sorrendben legyenek rendezettek!

| kategoria | max_brutto |
|-----------|--------------------|
| laptop | 2057.5099958777428 |
| mobil | 808.0099983811378 |
| tv | 593.8099988102913 |

A termékek tábla

| id | nev | kategoria | netto | penznem | afa |
|----|--------------|-----------|-------------|-------------|-------------|
| 1 | 4K TV | tv | 499 | EUR | 0.19 |
| 2 | Mobil 32GB | mobil | 299 | EUR | 0.19 |
| 3 | Mobil 128GB | mobil | 679 | EUR | 0.19 |
| 4 | Olcsó laptop | laptop | 269 | EUR | 0.19 |
| 5 | Drága laptop | laptop | 1729 | EUR | 0.19 |
| 6 | Könyv | könyv | <i>NULL</i> | <i>NULL</i> | <i>NULL</i> |

Tartalom I

15 Feltételek csoportosított mezőre (HAVING)

HAVING példa 1.

Melyik kategóriában mennyibe kerül a **legdrágább** termék?

```
SELECT `kategoria`, MAX(`netto`) AS legdragabb  
FROM `termekek`  
GROUP BY `kategoria`  
HAVING MAX(`netto`) is not null;
```

logikai hiba

| kategoria | legdragabb |
|-----------|------------|
| laptop | 1729 |
| mobil | 679 |
| tv | 499 |

- Ez nem az igazi! Itt a WHERE is elég lett volna!

HAVING példa 2.

Jelenítse meg a legalább két terméket tartalmazó kategóriákat!

MySQL

```
SELECT `kategoria`, COUNT(`id`) as `db`  
FROM `termekek`  
GROUP BY `kategoria`  
HAVING `db` >= 2;
```

| kategoria | db |
|-----------|----|
| laptop | 2 |
| mobil | 2 |

- Az nem volt kérdés, hogy hány termék van a kategóriában!

HAVING példa 3.

Jelenítse meg a legalább két kategóriát tartalmazó termékeket!

MySQL

```
SELECT `kategoria`  
FROM `termek`  
GROUP BY `kategoria`  
HAVING COUNT(`id`) >= 2;
```

| kategoria |
|-----------|
|-----------|

| |
|--------|
| laptop |
|--------|

| |
|-------|
| mobil |
|-------|

Tartalom I

16 Többtáblás lekérdezések

- Descartes szorzat
- INNER JOIN
- OUTER JOIN

Tartalom

16 Töbبتáblás lekérdezések

- Descartes szorzat
- INNER JOIN
- OUTER JOIN

Példa: a felhasználó és a cikk tábla

A két tábla az alábbi adatokat tartalmazza:

| felhasznalo | |
|-------------|-------|
| id | nev |
| 1 | Norbi |
| 2 | Bea |
| 3 | Helga |

| cikk | | |
|------|----------------|----------------------|
| id | felhasznalo_id | cim |
| 1 | 1 | Első cikk |
| 2 | 3 | Új motorom |
| 3 | 3 | Hogyan lettem videós |
| 4 | 1 | Új nap kezdődik |

Példa: A descartes szorzat eredménye. (*felhasznalo* \times *cikk*)

| id | nev | id | felhasznalo_id | cim |
|----------|--------------|----------|----------------|-----------------------------|
| 1 | Norbi | 1 | 1 | Első cikk |
| 1 | Norbi | 2 | 3 | Új motorom |
| 1 | Norbi | 3 | 3 | Hogyan lettem videós |
| 1 | Norbi | 4 | 1 | Új nap kezdődik |
| 2 | Bea | 1 | 1 | Első cikk |
| 2 | Bea | 2 | 3 | Új motorom |
| 2 | Bea | 3 | 3 | Hogyan lettem videós |
| 2 | Bea | 4 | 1 | Új nap kezdődik |
| 3 | Helga | 1 | 1 | Első cikk |
| 3 | Helga | 2 | 3 | Új motorom |
| 3 | Helga | 3 | 3 | Hogyan lettem videós |
| 3 | Helga | 4 | 1 | Új nap kezdődik |

Descartes szorzat (direkt szorzat) $A \times B$

Amennyiben a FROM után több táblát is megadunk vesszővel, akkor a lekérdezés során a táblák descartes szorzatát kapjuk.

```
SELECT * FROM `felhasznalo`, `cikk`;
```

MySQL

Probléma: Több hamis sor is keletkezett.

Valós adatok kinyerése a descartes szorzatból

MySQL

```
SELECT * FROM `felhasznalo`, `cikk`  
WHERE `felhasznalo`.`id` = `cikk`.`felhasznalo_id`;
```

| id | nev | id | felhasznalo_id | cim |
|----|-------|----|----------------|----------------------|
| 1 | Norbi | 1 | 1 | Első cikk |
| 1 | Norbi | 4 | 1 | Új nap kezdődik |
| 3 | Helga | 2 | 3 | Új motorom |
| 3 | Helga | 3 | 3 | Hogyan lettem videós |

Figyelem!

A **felhasznalo.id** (ponttal) és a **felhasznalo_id** (aláhúzással) nem összekeverendő!

Tartalom

16 Töbبتáblás lekrdézések

- Descartes szorzat
- **INNER JOIN**
- OUTER JOIN

INNER JOIN (2 tábla)

Két táblás lekérdezés:

```
SELECT * FROM `t1`  
    [INNER] JOIN `t2`  
        ON `t1`.`id` = `t2`.`t1_id`;
```

MySQL

```
SELECT * FROM `felhasznalo`  
    INNER JOIN `cikk`  
        ON `felhasznalo`.`id` = `cikk`.`felhasznalo_id`;
```

Figyelem!

Belső összekapcsoláskor csak azok a sorok jelennek meg, ahol van összeköthető adat a két táblában!

INNER JOIN (3 tábla)

Három táblás lekérdezés

```
SELECT * FROM `t1`  
    [INNER] JOIN `t2`  
        ON `t1`.`id` = `t2`.`t1_id`;  
    [INNER] JOIN `t3`  
        ON `t2`.`id` = `t3`.`t2_id`;
```


Tartalom

16 Töbبتáblás lekrdézések

- Descartes szorzat
- INNER JOIN
- OUTER JOIN

INNER JOIN vs. OUTER JOIN

A külső és belső összekapcsolások között annyi a különbség, hogy amíg a belső összekapcsolásnál csak azok a sorok jelennek meg, ahol mind a két táblában van összeköthető adat, addig a külső összekapcsolásnál elég ha valamelyik oldalon van adat.

Ez persze függ attól, hogy LEFT, RIGHT vagy FULL JOIN-ről van szó.

Azon sorokhoz, melyekhez nem lehet adatot találni a másik táblában, ott a hiányzó részek NULL értékekkel lesz kitöltve.

A MySQL **nem támogatja** a FULL OUTER JOIN-t.

LEFT OUTER JOIN

A **t1** tábla **minden** sorát megjeleníti, ahol tudott hozzá adatot találni a t2-ből ott megjelenik, ahol nem, ott NULL értékek lesznek.

```
SELECT * FROM t1
  LEFT [OUTER] JOIN
    t2 ON t1.id = t2.t1_id;
```

Példa 1: LEFT OUTER JOIN

Ki nem írt még cikket a blogra?

MySQL

```
SELECT *  
FROM  
    `felhasznalo`  
    LEFT OUTER JOIN `cikk`  
        ON `felhasznalo`.`id` = `cikk`.`felhasznalo_id`  
WHERE `cikk`.`felhasznalo_id` IS NULL;
```

Példa 1: LEFT OUTER JOIN

Ki nem írt még cikket a blogra?

| id | nev | id | felhasznalo_id | cim |
|----------|------------|----|----------------|----------------------|
| 1 | Norbi | 1 | 1 | Első cikk |
| 1 | Norbi | 4 | 1 | Új nap kezdődik |
| 2 | Bea | - | - | - |
| 3 | Helga | 2 | 3 | Új motorom |
| 3 | Helga | 3 | 3 | Hogyan lettem videós |

Bea még nem írt cikket, így a felhasznalo_id és a cim mezőben NULL értékek szerepelnek a lekérdezés eredményében.

| |
|-----|
| nev |
|-----|

| |
|-----|
| Bea |
|-----|

RIGHT OUTER JOIN

A **t2** tábla **minden** sorát megjeleníti, ahol tudott hozzá adatot találni a **t1**-ből ott megjelenik, ahol nem, ott NULL értékek lesznek.

```
SELECT * FROM t1  
RIGHT OUTER JOIN t2 ON t1.id = t2.t1_id;
```

Példa: RIGHT OUTER JOIN

Melyik felhasználó nem írt még cikket a blogra?

MySQL

```
SELECT *  
FROM  
    `cikk`  
    RIGHT OUTER JOIN `felhasznalo`  
        ON `felhasznalo`.`id` = `cikk`.`felhasznalo_id`  
WHERE `cikk`.`felhasznalo_id` IS NULL;
```

Példa: RIGHT OUTER JOIN

Ki nem írt még cikket a blogra?

| id | felhasznalo_id | cim | id | nev |
|----|----------------|----------------------|----------|------------|
| 1 | 1 | Első cikk | 1 | Norbi |
| 4 | 1 | Új nap kezdődik | 1 | Norbi |
| - | - | - | 2 | Bea |
| 2 | 3 | Új motorom | 3 | Helga |
| 3 | 3 | Hogyan lettem videós | 3 | Helga |

Bea még nem írt cikket, így a felhasznalo_id és a cim mezőben NULL értékek szerepelnek a lekérdezés eredményében.

| |
|-----|
| nev |
|-----|

| |
|-----|
| Bea |
|-----|

Férj és feleség táblák

| ferj | | |
|------|-------|----------|
| id | nev | felesege |
| 1 | Tamás | - |
| 2 | Laci | 3 |
| 3 | Peti | 1 |

| feleseg | | |
|---------|--------|-------|
| id | nev | ferje |
| 1 | Andrea | 3 |
| 2 | Emese | - |
| 3 | Nóra | 2 |

Feltételezzük, a monogám kapcsolati viszonyt, így a táblák között 1:1 kapcsolat áll fenn.

Figyelem!

Az itt látható táblák tervezése nem megfelelő, de a bemutatni kívánt anyagrész megértését elősegíti. Senki se próbálja ki otthon!

feleseg LEFT OUTER JOIN ferj

MySQL

```
SELECT * FROM `feleseg`  
  LEFT OUTER JOIN `ferj`  
    ON `feleseg`.`ferje` = `ferj`.`id`;
```

| id | nev | ferje | id | nev | felesege |
|----|--------|-------|----|------|----------|
| 1 | Andrea | 3 | 3 | Peti | 1 |
| 2 | Emese | - | - | - | - |
| 3 | Nóra | 2 | 2 | Laci | 3 |

Az összes feleség felsorolásra kerül, még az is, akinek nincs férje, csak utóbbinál NULL értékek szerepelnek a férj helyén.

ferj LEFT OUTER JOIN feleseg

MySQL

```
SELECT * FROM `ferj`  
  LEFT OUTER JOIN `felesege`  
    ON `ferj`.`felesege` = `felesege`.`id`;
```

| id | nev | felesege | id | nev | ferje |
|----|-------|----------|----|--------|-------|
| 1 | Tamás | - | - | - | - |
| 2 | Laci | 3 | 3 | Nóra | 2 |
| 3 | Peti | 1 | 1 | Andrea | 3 |

Az összes férj felsorolásra kerül, még az is, akinek nincs felesége, csak utóbbinál NULL értékek szerepelnek a feleség helyén.

feleseg RIGHT OUTER JOIN ferj

MySQL

```
SELECT * FROM `feleseg`  
  RIGHT OUTER JOIN `ferj`  
    ON `feleseg`.`ferje` = `ferj`.`id` ;
```

| id | nev | ferje | id | nev | felesege |
|----|--------|-------|----|-------|----------|
| - | - | - | 1 | Tamás | - |
| 3 | Nóra | 2 | 2 | Laci | 3 |
| 1 | Andrea | 3 | 3 | Peti | 1 |

Az összes férj megjelenik, még az is, akinek nincs felesége. Akinek van felesége, annak a felesége is megjelenik, akinek nincs, ott NULL értékek szerepelnek..

ferj RIGHT OUTER JOIN feleseg

MySQL

```
SELECT * FROM `ferj`  
  RIGHT OUTER JOIN `feleseg`  
    ON `ferj`.`felesege` = `feleseg`.`id`;
```

| id | nev | felesege | id | nev | ferje |
|----|------|----------|----|--------|-------|
| 3 | Peti | 1 | 1 | Andrea | 3 |
| - | - | - | 2 | Emese | - |
| 2 | Laci | 3 | 3 | Nóra | 2 |

Az összes feleség megjelenik, még az is, akinek nincs férje. Akinek van férje, annak a férje is megjelenik, akinek nincs, ott NULL értékek szerepelnek..

Tartalom I

17 Típusok

- Bevezető
- Szöveges típusok
- Numerikus típusok
- Dátum és idő típusok
- Speciális típusok

Tartalom

17 Típusok

- Bevezető
- Szöveges típusok
- Numerikus típusok
- Dátum és idő típusok
- Speciális típusok

Gyakori típusok

CHAR Fix hosszúságú karakterlánc

VARCHAR Változó hosszúságú karakterlánc

TEXT Nagy mennyiségű szöveg

INT Egész szám

FLOAT Lebegőpontos szám

DOUBLE Dupla pontosságú lebegőpontos szám

DATE Dátum

TIME Idő

DATETIME dátum és idő

Típusok linkek

Számok <https://dev.mysql.com/doc/refman/8.0/en/numeric-types.html>

Egész számok <https://dev.mysql.com/doc/refman/8.0/en/integer-types.html>

Lebegőpontos_számkok
<https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html>

Dátum és Idő <https://dev.mysql.com/doc/refman/8.0/en/date-and-time-types.html>

Szöveg <https://dev.mysql.com/doc/refman/8.0/en/string-types.html>

Tartalom

17 Típusok

- Bevezető
- Szöveges típusok
- Numerikus típusok
- Dátum és idő típusok
- Speciális típusok

CHAR(n)

- (fix hosszúságú) karakter
- Megadott darabszámú (n) karaktert tud eltárolni.
- Ez a méret 0 és 255 között mozoghat.
- Az eltárolt adatok hossza fix.
- Rövidebb szövegnél kitölti szóközökkel.
 - A PAD_CHAR_TO_FULL_LENGTH paraméter adja meg, hogy az adatok kiolvasásakor megjelenjenek vagy sem a kitöltő szóközök
- Túl hosszú szövegnél adatvesztés történik.

VARCHAR(n)

- Változó hosszúságú karakter
- Legfeljebb megadott darabszámú (n) karaktert tud eltárolni.
- Ez a méret 0 és 65 535¹ között van.
- El kell az adat mellett azt is tárolni, hogy hány karakterből áll.
- Rövidebb szövegnél nem történik semmi.
- Túl hosszú szövegnél adatvesztés történik!

¹Függ a karakterkészlettől és hogy egy sor mekkora adatot tárolhat

TEXT(M)

- Nagyobb szövegestartalom tárolására alkalmas, például egy üzenet vagy egy blogbejegyzés.
- Különböző méretek:
 - TINYTEXT 255 karakter
 - TEXT 65 535 karakter
 - MEDIUMTEXT 16 777 215 karakter
 - LONGTEXT 4 294 967 295 karakter (4 GB)
- Az **m** megadásakor a megfelelő méretet választja ki.
- <https://dev.mysql.com/doc/refman/8.0/en/string-type-syntax.html>

Tartalom

17 Típusok

- Bevezető
- Szöveges típusok
- **Numerikus típusok**
- Dátum és idő típusok
- Speciális típusok

INT(N)

- Előjeles egész szám tárolása

| Előtag | | minimum | maximum |
|--------|---|----------------------|---------------------|
| TINY | 1 | -128 | 127 |
| SMALL | 2 | -32768 | 32767 |
| MEDIUM | 3 | -8388608 | 8388607 |
| | 4 | -2147483648 | 2147483647 |
| BIG | 8 | -9223372036854775808 | 9223372036854775807 |

UNSIGNED

- Előjel nélküli egész számok
- Negatív szám nem írható be
- Cserébe kétszer annyi pozitív számunk lesz

| | | maximum |
|--------------------|---|----------------------|
| TINYINT UNSIGNED | 1 | 255 |
| SMALLINT UNSIGNED | 2 | 65535 |
| MEDIUMINT UNSIGNED | 3 | 16777215 |
| INT UNSIGNED | 4 | 4294967295 |
| BIGINT UNSIGNED | 8 | 18446744073709551615 |

FLOAT(m,d)

Lebegőpontos szám tárolására alkalmas típus.

m a karakterek száma

d a tizedesek száma

FLOAT(7,4) → 777,1234

- Összesen 7 számjegy jelenik meg, ebből 3 az egész rész, 4 a törtrész
- Az M és N értékének megadása nem kötelező, mivel nem SQL szabvány. Elhagyásával a kapott kód hordozhatóbb a különböző típusú adatbázisok között.

<https://dev.mysql.com/doc/refman/8.0/en/floating-point-types.html>

Tartalom

17 Típusok

- Bevezető
- Szöveges típusok
- Numerikus típusok
- Dátum és idő típusok
- Speciális típusok

DATE

- '1000-01-01' és '9999-12-31' közti értékeket tud eltárolni.

DATETIME

- '1000-01-01 00:00:00' és '9999-12-31 23:59:59' közti értékeket tud eltárolni.

TIMESTAMP

- '1970-01-01 00:00:01' UTC és '2038-01-19 03:14:07' UTC közti dátumot és időt tud eltárolni.
- Időzóna konverziót végez
- 2038 Probléma!

TIME

- '-838:59:59' és '838:59:59' közötti időértékeket tud eltárolni.
- <https://dev.mysql.com/doc/refman/5.5/en/time.html>

Tartalom

17 Típusok

- Bevezető
- Szöveges típusok
- Numerikus típusok
- Dátum és idő típusok
- Speciális típusok

ENUM

- Az enum egy speciális felsoroló típus
- Csak a megadott értékeket veheti fel az adott mező
- Az adat amit megadunk az szöveg, de a háttérben számként tárolja el
- A lekérdezés könnyen olvasható marad

MySQL

```
`jegy_tipus` ENUM('hazi','dolgozat','temazaro')
```

- Ekkor a **jegy_tipus** mezőben csak a felsorolt értékek lehetnek.
- Feltételben:

MySQL

```
WHERE `jegy_tipus` = 'temazaro'
```


BOOL/BOOLEAN

- Valójában az eltárolt adat típusa TINYINT(1) lesz.
- Az 1 IGAZ
- minden más HAMIS

Tartalom I

18 Adatbázis létrehozása és törlése

- Adatbázis létrehozása
- Alapértelmezett adatbázis kiválasztása
- Adatbázis törlése

Tartalom

18 Adatbázis létrehozása és törlése

- Adatbázis létrehozása
- Alapértelmezett adatbázis kiválasztása
- Adatbázis törlése

Adatbázis létrehozása (minimális példa)

MySQL

```
CREATE DATABASE `youtuberek` ;
```

- A legegyszerűbb parancs adatbázis létrehozásához.
- A karakterkódolás a szerver alapértelmezett beállítása lesz.
 - például: latin1_swedish

Adatbázis létrehozása karakterkódolás megadásával

MySQL

```
CREATE DATABASE `youtuberek`  
CHARACTER SET utf8  
COLLATE utf8_hungarian_ci;
```

- Adatbázis neve: **youtuberek**
- Karakterkódolás: **utf8**
- Nyelvi beállítások: **utf8_hungarian_ci**
 - Rendezés: **Magyar**
 - Kis és nagybetűk: érzéketlen (**Case Insensitive**)

Adatbázis létrehozása (ha még nem létezik a tábla)

MySQL

```
CREATE DATABASE IF NOT EXISTS `youtuberek`  
CHARACTER SET utf8  
COLLATE utf8_hungarian_ci;
```

- Csak akkor hozza létre az adatbázist, ha még nem létezik.
- Ezzel az „adatbázis már létezik” hibaüzenet kiküszöbölhető egy script futtatásakor.

Tartalom

18 Adatbázis létrehozása és törlése

- Adatbázis létrehozása
- Alapértelmezett adatbázis kiválasztása
- Adatbázis törlése

Alapértelmezett adatbázis kiválasztása

MySQL

```
USE `youtuberek`;
```

- Mivel még csak most hoztuk létre az adatbázist, így a use paranccsal megadhatjuk, hogy a lekérdezésekhez ezentúl ezt szeretnénk használni.

Tartalom

18 Adatbázis létrehozása és törlése

- Adatbázis létrehozása
- Alapértelmezett adatbázis kiválasztása
- Adatbázis törlése

Adatbázis törlése - DROP DATABASE

```
DROP DATABASE | SCHEMA [IF EXISTS] db_name;
```

- A **drop database** segítségével dobhatjuk el az adatbázist.
- Az adatbázisban tárolt adatok véglegesen elvesznek!

További olvasnivaló:

<https://dev.mysql.com/doc/refman/8.0/en/drop-database.html>

Adatbázis törlése

MySQL

```
DROP DATABASE `youtuberek`;
```

- A **drop database** segítségével dobhatjuk el az adatbázist.
- Az adatbázisban tárolt adatok véglegesen elvesznek!

Adatbázis törlés (csak ha létezik)

MySQL

```
DROP DATABASE IF EXISTS `youtuberek`
```

- A **drop database** segítségével dobhatjuk el az adatbázist.
- Az adatbázisban tárolt adatok véglegesen elvesznek!

Tartalom I

19 Tábla létrehozása és törlése

- Tábla létrehozása
- Opcionális mezők
- Kötelező mezők
- Elsődleges kulcs megadása a tábla létrehozásakor
- Tábla törlése

Tartalom

19 Tábla létrehozása és törlése

- Tábla létrehozása
- Opcionális mezők
- Kötelező mezők
- Elsődleges kulcs megadása a tábla létrehozásakor
- Tábla törlése

Tábla létrehozása - CREATE TABLE

```
CREATE TABLE `tabla_neve` (  
    `mezo` tipus1 egyebek1,  
    `mezo2` tipus2 egyebek2  
);
```

- A **tabla_neve** tetszőleges, általunk választható!
- - Célszerű eldönteni, hogy egyes vagy többesszámot alkalmazunk!
 - Mind a kettő lehet jó megoldás, de legyen egységes!
- A mező nevek is tetszőlegesek.

További olvasnivaló:

<https://dev.mysql.com/doc/refman/8.0/en/create-table.html>

Elnevezési szabályok

- A választott név ne legyen kulcsszó!
- Ne tartalmazzanak szóközt!

Amennyiben mindenképpen kulcsszót szeretnénk alkalmazni névként, akkor azt határolók közé kell elhelyezni.

MySQL esetében ez a backtick lesz.

További olvasnivaló:

<https://dev.mysql.com/doc/refman/8.0/en/keywords.html>

Tartalom

19 Tábla létrehozása és törlése

- Tábla létrehozása
- Opcionális mezők
- Kötelező mezők
- Elsődleges kulcs megadása a tábla létrehozásakor
- Tábla törlése

Opcionális mezők, alapértelmezett értékek

- A "DEFAULT érték" kódrészlettel adhatjuk meg, hogy mit írjon az adott mezőbe, ha nem adnánk meg értéket.

MySQL

```
CREATE TABLE `pelda` (  
    `szin` VARCHAR(25) DEFAULT 'sárga',  
);
```

- Mondhatjuk, hogy legyen az alapértelmezett NULL, vagy elhagyhatjuk. Mind két esetben NULL-t fog beszúrni az adott mezőbe, amennyiben nem határozzunk meg értéket

MySQL

```
CREATE TABLE `pelda` (  
    `szin` VARCHAR(25) DEFAULT NULL,  
);
```

Tartalom

19 Tábla létrehozása és törlése

- Tábla létrehozása
- Opcionális mezők
- **Kötelező mezők**
- Elsődleges kulcs megadása a tábla létrehozásakor
- Tábla törlése

Kötelezően kitöltendő mezők

- A "NOT NULL" kódrészlet megadásával kötelezően kitöltendővé tehetünk egy adott mezőt.

MySQL

```
CREATE TABLE `pelda` (  
  `szin` VARCHAR(25) NOT NULL,  
);
```

- Ebben az esetben nem adhatjuk meg, hogy null az alapértelmezett érték.

Szintaktikai hiba

```
CREATE TABLE `pelda` (  
  `szin` VARCHAR(25) DEFAULT NULL NOT NULL,  
);
```

Tábla létrehozás példa: videósok

Hozzunk létre az alábbi leírás alapján egy táblát.

| | | | |
|-----|------------|-----------------------|-------------|
| kor | Egész | kötelezően kitöltendő | Videós kora |
| név | Szöveg(25) | kötelezően kitöltendő | Videós neve |

MySQL

```
CREATE TABLE `videosok` (  
  `kor` INT NOT NULL COMMENT 'Videós kora',  
  `nev` VARCHAR(25) NOT NULL COMMENT 'Videós neve'  
);
```

A **COMMENT** segítségével magunknak tárolhatunk el információkat az adott mezőről.

Tartalom

19 Tábla létrehozása és törlése

- Tábla létrehozása
- Opcionális mezők
- Kötelező mezők
- Elsődleges kulcs megadása a tábla létrehozásakor
- Tábla törlése

Elsődleges kulcs

- Az elsődleges kulcs egyértelműen meghatároz egy rekordot (sort) a táblában.
- Az elsődleges kulcs állhat több mezőből, ekkor **összetett kulcsról** beszélünk.
- Egy táblának csak egy elsődleges kulcsa lehet!
- Kitöltése kötelező! Amennyiben nem adjuk meg, hogy legyen NOT NULL ezt az adatbázis kezelő hozzáteszi a háttérben.

<https://dev.mysql.com/doc/refman/8.0/en/create-table.html>

Elsődleges kulcs megadása

- Elsődleges kulcs megadás a mező tulajdonságként:

MySQL

```
CREATE TABLE `pelda` (  
  `id` INT NOT NULL PRIMARY KEY  
);
```

- Elsődleges kulcs megadás a mezők felsorolása után:

MySQL

```
CREATE TABLE `pelda` (  
  `id` INT NOT NULL,  
  PRIMARY KEY (`id`)  
);
```

A fenti két utasítás ugyanazt eredményezi!

Összetett (elsődleges) kulcs hibás példa

Az elsődleges kulcs állhat több mezőből, ekkor **összetett kulcs**ról beszélünk.

```
CREATE TABLE `pelda` (  
  `nev` VARCHAR(20) NOT NULL PRIMARY KEY,  
  `kor` INT(11) NOT NULL PRIMARY KEY,  
  `cim` VARCHAR(20)  
);
```

Szintaktikai hiba

Közvetlen tulajdonságként nem tudjuk megadni.

Többszörös elsődleges kulcs definiálás.

Hibakód: #1068

Összetett (elsődleges) kulcs helyes példa

A mezők felsorolása után adhatjuk meg

MySQL

```
CREATE TABLE `pelda` (  
  `nev` VARCHAR(20) NOT NULL,  
  `kor` INT(11) NOT NULL,  
  `cim` VARCHAR(20),  
  PRIMARY KEY (`nev` , `kor`)  
);
```

ID mező példa

MySQL

```
CREATE TABLE `tantargy` (  
  `id` bigint unsigned NOT NULL AUTO_INCREMENT,  
  `nev` VARCHAR(30) NOT NULL,  
  PRIMARY KEY (id)  
);
```

- **bigint**: $2^{63} - 1$ szám tárolására alkalmas
- **unsigned**: nem lehet negatív
- **NOT NULL**: az elsődleges kulcs kitöltése kötelező
- **AUTO_INCREMENT**: az adott mező egy automatikusan növekvő szám

Az id alapértelmezett értékét a különböző adatbázis-kezelő rendszerekben máshogy kell beállítani:

https://www.w3schools.com/sql/sql_autoincrement.asp

Tartalom

19 Tábla létrehozása és törlése

- Tábla létrehozása
- Opcionális mezők
- Kötelező mezők
- Elsődleges kulcs megadása a tábla létrehozásakor
- Tábla törlése

Táblák törlése

- Egy tábla és adatainak törlése:

```
DROP TABLE `videosok`;
```

MySQL

- Egy tábla és adatainak törlése, ha létezik:

```
DROP TABLE IF EXISTS `videosok`;
```

MySQL

- Több tábla törlése az adatokkal együtt:

```
DROP TABLE `videosok`, `epizodok`;
```

MySQL

Tartalom I

- 20 Adatok beszúrása és törlése
 - Beszúrás (INSERT)
 - Tábla összes adatának törlése (TRUNCATE)
 - Adat törlése (DELETE)

Tartalom

- 20 Adatok beszúrása és törlése
 - Beszúrás (INSERT)
 - Tábla összes adatának törlése (TRUNCATE)
 - Adat törlése (DELETE)

Beszúrás minta

Új sor beszúrásánál meg kell adni, hogy melyik táblába szeretnénk beszúrni. Megadhatjuk, hogy mely mezőkbe szeretnénk adatot beszúrni, a VALUES() részbe pedig a beszúrandó értékek kerülnek.

MySQL

```
INSERT INTO `youtuberek` (`nev`, `kor`)  
VALUES('Horváth András', 44);
```


HIBA 1241

Gyakori hiba a fölösleges, hibás zárójelezés!

```
INSERT INTO `youtuberek`  
  (`nev`, `kor`)  
VALUES(  
  ('Szirami Gergely', 29),  
  ('Horváth András', 31)  
);
```

Szintaktikai hiba

Több sor beszúrása helyesen

MySQL

```
INSERT INTO `youtuberek`  
  (`nev`, `kor`)  
VALUES  
  ('Szirmai Gergely', 29),  
  ('Dancsó Péter', 31);
```

- Több sor beszúrásakor a VALUES() részt elég egyszer feltüntetni!

Tartalom

- 20 Adatok beszúrása és törlése
 - Beszúrás (INSERT)
 - Tábla összes adatának törlése (TRUNCATE)
 - Adat törlése (DELETE)

Tábla tartalmának a törlése

```
TRUNCATE [TABLE] tabla_neve;
```

Amennyiben a táblára még szükségünk van, de az adatokra nem, akkor kiüríthetjük a tartalmát.

Kifejezetten hasznos INSERT gyakorláskor elkövetett hibák javítására.

```
TRUNCATE TABLE `videosok`;
```

MySQL

Figyelem!

Az adatok törlése végleges!

Tartalom

- 20 Adatok beszúrása és törlése
 - Beszúrás (INSERT)
 - Tábla összes adatának törlése (TRUNCATE)
 - Adat törlése (DELETE)

DELETE

MySQL

```
DELETE FROM `dolgozo`;
```

- Kérdés nélkül töröl minden adatot a táblából **véglegesen**

MySQL

```
DELETE FROM `dolgozo`  
WHERE `nev` = 'Zója';
```

- Kérdés nélkül törli az összes Zója nevű dolgozót **véglegesen**

MySQL

```
DELETE FROM `dolgozo`  
WHERE `fizetes` > 10000000;
```

- Kérdés nélkül törli az összes 10 millió felett kereső dolgozót **véglegesen**

Tartalom I

21 Anomáliák

- Beszúrási anomália
- Módosítási anomália
- Törlési anomália

Fogalmalmi különbségek

Adatbázis tervezés során

- a **tábla** helyett **reláció**
- a **mező** helyett **attribútum**

került alkalmazásra

Anomáliák

Definition (Anomália fogalma)

Az anomália az adatbázisban olyan rendellenesség, mely valamely karbantartási műveletnél plusz műveletek beiktatását igényli.

- Beszúrási anomália
- Módosítási anomália
- Törlési anomália

Redundancia

Definition (Redundancia)

Redundanciáról akkor beszélünk, ha valamely tényt vagy a többi adatból levezethető mennyiséget ismételten (többszörösen) tároljuk az adatbázisban.

- Lehet hasznos adatbiztonság szempontjából, például biztonsági mentés
- A fölösleges ismétlődést célszerű elkerülni
- Az idegen kulcs — bár redundánsan jelenik meg — a hozzá kapcsolódó adatokat takarja együttesen, így végeredményben csökkenti a redundanciát

Tartalom

21 Anomáliák

- Beszúrási anomália
- Módosítási anomália
- Törlési anomália

Beszúrási anomália

Definition (Beszúrási anomália)

Beszúrási anomáliáról beszélünk abban az esetben, amikor egy adatrekord beszúrása egy másik, hozzá logikailag nem kapcsolódó adatcsoport beszúrást kívánja meg.

Beszúrási anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 1 | 47 | lakás | 86 0000 | Péter | 1600 |
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4300 |

- Az ingatlanokhoz logikailag nem kapcsolódik az ingatlanos fizetése
- Az, hogy ki árulja az ingatlant egy hasznos információ
- Mariann fizetése redundánsan szerepel a táblában

Beszúrási anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 1 | 47 | lakás | 86 0000 | Péter | 1600 |
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4300 |
| 5 | 68 | lakás | 1 050 000 | Mariann | 4300 |

- Az ingatlanokhoz logikailag nem kapcsolódik az ingatlanos fizetése
- Az, hogy ki árulja az ingatlant egy hasznos információ
- Mariann fizetése redundánsan szerepel a táblában
- **Mariann fizetését is meg kell adni egy ingatlan új felviteléhez**

Tartalom

21 Anomáliák

- Beszúrási anomália
- **Módosítási anomália**
- Törlési anomália

Módosítási anomália

Definition (Módosítási anomália)

Abban az esetben, ha egy relációban egy adat módosítása több helyen történő módosítást igényel, akkor módosítási anomáliáról beszélünk.

Módosítási anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 1 | 47 | lakás | 86 0000 | Péter | 1600 |
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4300 |
| 5 | 68 | lakás | 1 050 000 | Mariann | 4300 |

- A 4-es azonosítójú ingatlan eladása miatt jár 200 euró fizetésemelés az ingatlanosának

Módosítási anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 1 | 47 | lakás | 86 0000 | Péter | 1600 |
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4500 |
| 5 | 68 | lakás | 1 050 000 | Mariann | 4300 |

- A 4-es azonosítójú ingatlan eladása miatt jár 200 euró fizetésemelés az ingatlanosának

Módosítási anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 1 | 47 | lakás | 86 0000 | Péter | 1600 |
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4500 |
| 5 | 68 | lakás | 1 050 000 | Mariann | 4300 |

- Mivel az ingatlanos fizetése redundánsan van eltárolva, így **plusz műveleteket kell elvégezni**, a többi érték módosításához
- Ha minden "Mariann" nevű ingatlanos fizetését emeljük, akkor előfordulhatna, hogy egy **másik** személy kap emelést

Tartalom

21 Anomáliák

- Beszúrási anomália
- Módosítási anomália
- Törlési anomália

Törlési anomália

Definition (Törlési anomália)

Amennyiben egy adat törlésével másik, hozzá logikailag nem kapcsolódó adatcsoportot is elveszítünk, törlési anomáliáról beszélünk.

Törlési anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 1 | 47 | lakás | 86 0000 | Péter | 1600 |
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4300 |
| 5 | 68 | lakás | 1 050 000 | Mariann | 4300 |

- A 1-es azonosítójú ingatlan tulajdonosával szerződést bontott az ügynökség, kerüljön törlésre

Törlési anomália példa

| `id` | `meret` | `tipus` | `ar` | `ingatlanos` | `fizetes` |
|------|---------|---------|-----------|--------------|-----------|
| 2 | 47 | ház | 1 495 000 | Mariann | 4300 |
| 3 | 55 | lakás | 990 000 | Zoli | 6500 |
| 4 | 214 | ház | 1 250 000 | Mariann | 4300 |
| 5 | 68 | lakás | 1 050 000 | Mariann | 4300 |

- A 1-es azonosítójú ingatlan tulajdonosával szerződést bontott az ügynökség, kerüljön törlésre
- Mennyi Péter fizetése?
- **Az ingatlan törlésével egy hozzá logikailag nem kapcsolódó adat (Péter fizetése) is törlésre került**

Tartalom I

22 Kulcsok

- Szuperkulcs
- Kulcs(jelölt)
- Elsődleges kulcs

Tartalom

22 Kulcsok

- Szuperkulcs
- Kulcs(jelölt)
- Elsődleges kulcs

Szuperkulcs

Definition (Szuperkulcs)

Szuperkulcsnak nevezzük azt az attribútumhalmazt, (melyek attribútumait együtt véve) egyértelműen meghatároz egy rekordot a relációban.

- Tartalmazhat olyan ("fölösleges") attribútumot, amit elhagyva is teljesíti a feltételeket

Example (*Személy* {személyi_szám, név, kor})

- Szuperkulcsok a *Személy* relációban
 - {személyi_szám} Önmagában elég a személyi_szám
 - {személyi_szám, név}
 - {személyi_szám, név, kor}
- Az alábbi attribútumhalmazok **nem** teljesítik a feltételt
 - {név}
 - {kor}
 - {név, kor}

Tartalom

22 Kulcsok

- Szuperkulcs
- Kulcs(jelölt)
- Elsődleges kulcs

Kulcs vagy Kulcsjelölt

Definition (Kulcs(jelölt))

Egy olyan (minimális) szuperkulcs, melynek bármely attribútumának eltávolítása után már nem szuperkulcs.

Example (*Autó* (rendszer, alvázszám, gyártó, típus, üzemanyag))

- A *rendszer* és az *alvázszám* önmagában is beazonosít egy autót

| | Attribútumhalmaz | Szuperkulcs | Kulcs(jelölt) |
|---|--------------------------------|-------------|---------------|
| 1 | {rendszer} | igen | igen |
| 2 | {alvázszám} | igen | igen |
| 3 | {rendszer, alvázszám} | igen | nem |
| 4 | {gyártó, üzemanyag} | nem | nem |
| 5 | {rendszer, típus, gyártó} | igen | nem |
| 6 | {alvázszám, gyártó, üzemanyag} | igen | nem |

Tartalom

22 Kulcsok

- Szuperkulcs
- Kulcs(jelölt)
- Elsődleges kulcs

Elsődleges kulcs

- Az elsődleges kulcs
 - a kulcsjelöltek egyike lesz
 - az adatbázis tervezés során tetszőlegesen megválasztható
 - a relációban aláhúzással jelölendő
 - nem tartalmazhat ismétlődést
 - nem tartalmazhat NULL értéket
 - lehetőség szerint ne változzon meg, ha mégis akkor ez ne legyen gyakori
 - összetett kulcs helyett legyen egyszerű

Tartalom I

23 Normalizáció

- Funkcionális függőség

Normalizáció

Definition (Normalizáció)

A normalizáció egy olyan adatbázis tervezési technika, ami csökkenti a redundanciát, elősegíti az anomáliák kiküszöbölését. A nagy táblákat több kicsire bontja és kapcsolatokat határoz meg.

Definition (Normálforma)

Adatbázis tervezés folyamatában többnyire egymásra épülő szabályok rendszerének egy eleme.

Tartalom

- 23 Normalizáció
 - Funkcionális függőség

Funkcionális függőség

$$R(A_1, A_2, A_3, \dots, B_1, B_2, \dots, B_n)$$

Definition (Funkcionális függőség)

Egy adott R relációban, egy B tartomány funkcionálisan függ az A tartománytól, ha bármely időpontban, minden egyes A értékhez egyetlen B érték tartozik az adott reláción belül.

Másképp megfogalmazva: Az A attribútumhalmaz értékei egyértelműen meghatározzák a B attribútumhalmaz értékeit.

Funkcionális függőség: Személyi szám és név kapcsolata

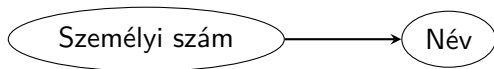
Vegyük a *Személy* (személyi_száma, név) relációt

A funkcionális függőség jelölése:

$$\{\text{személyi_szám}\} \rightarrow \{\text{név}\}$$

- Az attribútumhalmazokat kapcsolószerűjelekkel jelöljük
- A nyíl határozza meg a függés irányát, azaz a baloldali attribútum(ok) halmazától függ a jobb oldali attribútum(ok) halmaza

A funkcionális függőség vizualizálása:



- A személyi szám egyértelműen meghatározza a személy nevét
- Fordítva nem igaz, hiszen több embernek is lehet ugyanaz a neve, de a személyi számuk eltérő lesz

Funkcionális függőség: Személyi szám és név kapcsolata

A funkcionális függőségnek azért ez a neve, mert elméletben készíthetnénk olyan függvényt, ami tetszőleges, típusának megfelelő bemeneti adat esetén egyértelmű kimenetet eredményez.

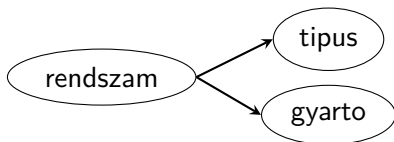
Python

```
def szemszambol_nev(szemszam):  
    if szemszam == "472278ZC":  
        return "Szentessy Péter"  
    elif szemszam == "258352DT":  
        return "Vasvári Mónika"  
    elif szemszam == "162633CX":  
        return "Vasvári Mónika"  
    ...
```

Funkcionális függőség példa: Autó

Vegyük az *Autó* (rendszer, gyártó, típus) relációt

| rendszer | gyártó | típus |
|----------|---------|-------|
| ABC-123 | Opel | Astra |
| DEF-444 | Vauxhal | Astra |
| PHP-404 | VW | Jetta |
| ASD-365 | VW | Polo |

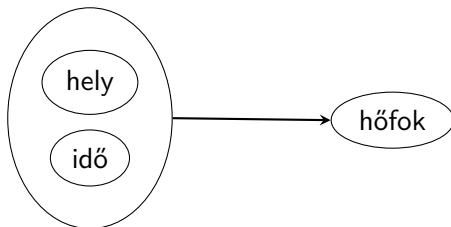


- $\{\text{rendszer}\} \rightarrow \{\text{típus}\}$
 - A rendszer meghatározza a típust
 - A típusból nem lehet a rendszert meghatározni (pl Astra)
- $\{\text{rendszer}\} \rightarrow \{\text{gyártó}\}$
 - A rendszer egyértelműen meghatározza a gyártót
 - A gyártóból nem határozható meg egyértelműen a rendszer (pl.: VW)
- $\{\text{rendszer}\} \rightarrow \{\text{típus, gyártó}\}$
 - A rendszer meghatározza a típust és a gyártót is. A fentieket magába foglalja, ez kell nekünk, ez látható az ábrán

Funkcionális függőség példa: Napi hőmérséklet

Magyarország egy nap alatt végrehajtott méréseit szeretnénk tárolni. Vegyük a *napi_hőmérséklet* (hely, időpont, hőmérséklet) relációt

| hely | idő | hőfok |
|------|-------|-------|
| Győr | 08:00 | 10 |
| Győr | 08:05 | 10 |
| Pécs | 08:00 | 10 |
| Pécs | 09:00 | 18 |



- A $\{\text{hely}, \text{időpont}\} \rightarrow \{\text{hőmérséklet}\}$ függőség teljesül
 - A hely **önmagában nem** határozza meg a hőfokot, mert ugyanazon a helyen, de másik időpontban lehet más a hőmérséklet
 - Az idő **önmagában nem** határozza meg a hőfokot, mert ugyanabban az időben lehet máshol más hőmérséklet
 - A hely és az idő **együttesen határozzák** meg a hőfokot

Triviális funkcionális függőség

Vegyük az *Autó* (rendszer, gyártó, típus) relációt

- $\{\text{típus}\} \rightarrow \{\text{típus}\}$
 - Saját magából megállapítható önmaga. Ez triviális
- $\{\text{típus, szín}\} \rightarrow \{\text{típus}\}$
 - Triviális egy funkcionális függőség, amennyiben a "bal oldali" halmaznak részhalmaza a "jobb oldali" attribútumhalmaz

Ezek a triviális függőségek a kivételektől eltekintve nem lesznek hasznunkra adatbázis tervezésnél, így a legtöbb esetben elhagyhatóak.

Funkcionális függőség: Személy

Vegyük a *Személy* (név, szül_idő, szül_hely, anyja_neve, cim, tel) relációt

| név | szül_idő | szül_hely | anyja_neve | cim | tel |
|----------|------------|-----------|------------|-------|---------|
| T. Péter | 1986-01-05 | Bp | K. Mária | Bp. | 1234569 |
| T. Péter | 1986-01-05 | Bp | M. Emese | Győr. | 5525359 |
| T. Péter | 1999-10-12 | Győr | K. Mária | Bp. | 1122339 |

• Függőségek

- {név, szül_idő, szül_hely, anyja_neve, **cim**, **tel**} → {**cim**, **tel**}
- {név, szül_idő, szül_hely, anyja_neve, **cim**} → {cim, **tel**}
- {név, szül_idő, szül_hely, anyja_neve, **tel**} → {cim, **tel**}
- {név, szül_idő, szül_hely, anyja_neve} → {cim, tel}

Funkcionális függőség: Személy

$$\{\text{név, szül_idő, szül_hely, anyja_neve}\} \rightarrow \{\text{cim, tel}\}$$

- A fenti függőség az összes mezőt tartalmazza
- A függőség bal oldala minimális kulcs, más nincs
- Könnyű egy egy elemű halmazból **elsődleges kulcsot** kinevezni
- Mivel több attribútumból tevődik össze, így **összetett kulcs** lesz

Funkcionális függőség: Személy

Vegyük az alábbi relációt

Személy (személyi_száma, név, szül_idő, szül_hely, anyja_neve, cím, tel)

| személyi_száma | név | szül_idő | szül_hely | anyja_neve | cím | tel |
|----------------|-------|------------|-----------|------------|-------|---------|
| 1-860105-7825 | T. P. | 1986-01-05 | Bp | K. Mária | Bp. | 1234569 |
| 1-860105-2353 | T. P. | 1986-01-05 | Bp | M. Emese | Győr. | 5525359 |
| 1-901012-4581 | T. P. | 1999-10-12 | Győr | K. Mária | Bp. | 1122339 |

- Nem triviális funkcionális függőségek:
 - {személyi_száma} \rightarrow {szül_idő}
 - {személyi_száma} \rightarrow {név, szül_idő, szül_hely, anyja_neve, cím, tel}
 - {név, szül_idő, szül_hely, anyja_neve} \rightarrow {cím, tel}

Funkcionális függőség: Személy

- Vizsgáljuk meg az alábbi függőségeket
 - $\{\text{név, szül_idő, szül_hely, anyja_neve}\} \rightarrow \{\text{szemelyi_szám, cím, tel}\}$
 - $\{\text{szemelyi_szám}\} \rightarrow \{\text{név, szül_idő, szül_hely, anyja_neve, cím, tel}\}$
- Mind a két függőségre igaz, hogy a reláció összes attribútuma szerepel valamelyik oldalán
- A $\{\text{név, szül_idő, szül_hely, anyja_neve}\}$ attribútumhalmaz egy olyan **szuperkulcs**, ami minimális, így **kulcs(jelölt)** is egyben
- A $\{\text{szemelyi_szám}\}$ attribútumhalmaz egy olyan **szuperkulcs**, ami minimális, így **kulcs(jelölt)** is egyben
- Válasszuk azt a kulcsjelöltet, amelyik
 - nem, vagy csak ritkán változik
 - egyszerű
- Mivel a $\{\text{szemelyi_szám}\}$ egy attribútumból áll, így az lesz az **egyszerű kulcs** lesz

Funkcionális függőség példa: Irányítószám és település

Vegyük az *Cim* (irsz, telepules) relációt

| irsz | település |
|------|------------------|
| 1015 | Budapest |
| 1033 | Budapest |
| 1213 | Budapest |
| 6710 | Szeged |
| 6726 | Szeged |
| 9400 | Sopron |
| 9444 | Fertőszentmiklós |

| irsz | település |
|------|----------------|
| 8984 | Gombosszeg |
| 8984 | Petrikeresztúr |
| 8984 | Iborfia |

- Megállapítható -e a település nevéből az irányítószám? Nem
- Megállapítható -e az irányítószámból a település neve? Nem
- $\{\text{irsz}\} \rightarrow \{\text{település}\}$
 - Klasszikus példa, **DE Magyarországra nem érvényes**
- Melyik település irányítószáma 1015? Bécs
 - Európai adatbázisban sem állja meg a helyét

Tartalom I

24 Adatbázis módosítása

Adatbázis módosítása

Amennyiben elfelejtettük beállíthatjuk utólag is a karakterkódolást, pontosabban módosíthatjuk. Ezt lehetőleg az adatok beszúrása előtt tegyük még meg.

MySQL

```
ALTER DATABASE `videosok`  
CHARACTER SET utf8  
COLLATE utf8_hungarian_ci;
```

- Karakterkészletek lekérdezése: **SHOW CHARACTER SET**
- Nyelvi beállítások lekérdezése: **SHOW COLLATION**

Tartalom I

- 25 Táblák módosítása
 - Új mező hozzáadása
 - Mező törlése
 - Mező módosítása
 - Tábla módosítása

Tábla módosítása - ALTER TABLE

```
ALTER TABLE tabla_nev  
modositasok_listaja;
```

- mező hozzáadása/törlése
- index hozzáadása/törlése
- kulcs hozzáadása/törlése
- megszorítás hozzáadása/törlése

További olvasnivaló:

<https://dev.mysql.com/doc/refman/8.0/en/alter-table.html>

https://en.wikibooks.org/wiki/Structured_Query_Language/Alter_Table

Tartalom

- 25 Táblák módosítása
 - Új mező hozzáadása
 - Mező törlése
 - Mező módosítása
 - Tábla módosítása

Mező hozzáadása - ADD COLUMN

```
ALTER TABLE tabla_nev  
ADD [COLUMN] mezo_neve mezo_definicio  
[FIRST | AFTER masik_mezo_neve]
```

- A **COLUMN** elhagyható.
- Megadhatjuk, hogy az új mező legyen az első **FIRST**, vagy
- a meghatározott másik mező után legyen **AFTER**

Mező hozzáadása példa

Adja hozzá a leírásnak megfelelő mezőt az **auto** táblához!

| | | |
|------|------------|-----------------------|
| szin | Szöveg(25) | kötelezően kitöltendő |
|------|------------|-----------------------|

```
ALTER TABLE auto  
  ADD COLUMN szin VARCHAR(25) NOT NULL;
```

MySQL

Mező hozzáadása (egyszerre több) példa

Adja hozzá a leírásnak megfelelő mezőket az **auto** táblához!

| | | |
|------------|------------|--------------------------|
| kor | Egész | alapértelmezett érték: 0 |
| tulajdonos | Szöveg(25) | kötelezően kitöltendő |

```
ALTER TABLE auto  
  ADD COLUMN kor INT DEFAULT 0,  
  ADD COLUMN tulajdonos VARCHAR(25) NOT NULL;
```

MySQL

Tartalom

- 25 Táblák módosítása
 - Új mező hozzáadása
 - Mező törlése
 - Mező módosítása
 - Tábla módosítása

Mező törlése - DROP COLUMN

```
ALTER TABLE tabla_nev  
DROP [COLUMN] mezo_neve;
```

- A **COLUMN** elhagyható.

Mező törlése példák

1. Törölje az **auto** táblából a **szin** mezőt!

```
ALTER TABLE auto  
  DROP COLUMN szin;
```

MySQL

2. Törölje az **auto** táblából a **kor** és a **tulajdonos** mezőket!

```
ALTER TABLE auto  
  DROP COLUMN kor,  
  DROP COLUMN tulajdonos;
```

MySQL

MySQL-ben a szabványtól eltérően egyszerre több mezőt is törölhetünk egy ALTER utasításban.

Szabvány

Tartalom

- 25 Táblák módosítása
 - Új mező hozzáadása
 - Mező törlése
 - **Mező módosítása**
 - Tábla módosítása

Mező módosítás - CHANGE COLUMN

```
ALTER TABLE tabla_nev  
CHANGE [COLUMN] regi_nev uj_nev típusdefiníció;
```

- A **COLUMN** elhagyható.
- Módosíthatjuk egyszerre a mező nevét és típusát is
- Ha a típusát nem szeretnénk módosítani, akkor ugyanazt kell itt is megadni, mint a jelenlegi típusa.
- Ha a nevét nem szeretnénk módosítani, akkor a **regi_nev** és az **uj_nev** helyére is ugyanazt kell írni, vagy mást (modify) használni!

Mező módosítása (CHANGE) példák

1. Módosítsa az **auto** táblában a **kategoria** mező nevét **kat**-ra!

MySQL

```
ALTER TABLE auto  
  CHANGE COLUMN `kategoria` `kat` VARCHAR(5);
```

2. Módosítsa az **auto** tábla **szin** mezőt úgy, hogy 20 karaktert lehessen eltárolni benne!

MySQL

```
ALTER TABLE auto  
  CHANGE COLUMN szin szin VARCHAR(20);
```

3. Tegye kötelezően kitöltendővé az **auto** tábla **szin** mezőjét!

MySQL

```
ALTER TABLE auto  
  CHANGE COLUMN szin szin VARCHAR(20) NOT NULL;
```

Mező módosítása - MODIFY COLUMN

```
ALTER TABLE tabla_nev  
MODIFY [COLUMN] mezo_neve tipusdefinicio;
```

- A **COLUMN** elhagyható.
- A **MODIFY** nem tudja a mező nevét módosítani!

Mező módosítása (MODIFY) példák

1. Módosítsa az **auto** tábla **szin** mezőjét úgy, hogy 20 karaktert lehessen eltárolni benne!

```
ALTER TABLE auto  
  MODIFY COLUMN szin VARCHAR(20);
```

MySQL

2. Tegye kötelezően kitöltendővé az **auto** tábla **szin** mezőjét!

```
ALTER TABLE auto  
  MODIFY COLUMN szin VARCHAR(20) NOT NULL;
```

MySQL

Amikor a mező nevét nem szeretnénk módosítani célszerűbb a **MODIFY**-t választani!

Tipp!

Összetett módosítások

Módosítsa az **auto** táblát az alábbiak szerint:

- módosítsa a **szin** mezőjét úgy, hogy 20 karaktert lehessen eltárolni benne!
- törölje a **tulajdonos** mezőjét
- adjon hozzá egy **kapcsolat** mezőt, amiben telefonszámot fogunk eltárolni

MySQL

```
ALTER TABLE auto
  MODIFY COLUMN szin VARCHAR(20),
  DROP COLUMN tulajdonos,
  ADD COLUMN `kapcsolat` VARCHAR(12)
;
```

Tábla mezőinek módosítása - összefoglaló

- **ALTER TABLE** Tábla (szerkezetének) módosítása
 - **ADD** Új mező hozzáadása
 - **DROP** Mező törlése
 - **CHANGE** név és típus módosítás
 - **MODIFY** csak típus módosítása

Tartalom

- 25 Táblák módosítása
 - Új mező hozzáadása
 - Mező törlése
 - Mező módosítása
 - Tábla módosítása

Tábla átnevezése

Nevezze át az **autok** táblát **jarmuvek** táblának.

```
RENAME TABLE `autok` `jarmuvek`;
```

MySQL

Tartalom I

26 Adatok módosítása (UPDATE)

Az UPDATE parancs

```
UPDATE [LOW_PRIORITY] [IGNORE] tabla  
SET assignment_list  
[WHERE where_condition]  
[ORDER BY ...]  
[LIMIT row_count]
```

value:

{expr | DEFAULT}

assignment:

col_name = value

assignment_list:

assignment [, assignment] ...

Update példa

Beosztás módosítása:

```
UPDATE t1 SET beosztas = 'alkalmazott';
```

MySQL

Ár növelése:

```
UPDATE t1 SET ar = ar * 1.2;
```

MySQL

Update példa

MySQL

```
UPDATE `tanulok`  
SET  
    `nev` = 'Kovács Petra'  
WHERE  
    `id` = '3472';
```

Update példa

ID növelése

```
UPDATE t SET id = id + 1 ORDER BY id DESC;
```

MySQL

A fenti példa csak akkor működik, ha rendezzük id szerint fordított sorrendben az adatokat!

Összetett update

- 1 SELECT lekérdezés írása
- 2 Átalakítás UPDATE lekérdezéssé.

Join vagy allekérdezés használható

Összetett update példa:

**A P. Sára nevű sofőr a PPU-832 rendszámú autót kapja meg!
Módosítsa ezek alapján az auto táblát!**

MySQL

```
UPDATE auto
SET
tulaj =
(SELECT id FROM személy WHERE nev = 'P. Sára')
WHERE
id =
(SELECT id FROM auto
WHERE rendszam = 'PPU-832' LIMIT 1);
```

Tartalom I

27 Megszorítások

- Elsődleges kulcs
- Idegen kulcs
- Egyedi értékek UNIQUE

MySQL CONSTRAINTS (megszorítások)

MySQL-ben többféle megszorítás is létezik:

- **DEFAULT**: Alapértelmezett érték megadása
- **NOT NULL**: Adott mező értékei nem vehetik fel a NULL-t
- **UNIQUE**: Az adott mezőben nem szerepelhet kétszer ugyanaz
- **PRIMARY KEY**: Elsődleges kulcs
- **FOREIGN KEY**: Idegen kulcs
- **CHECK**: Adat egyszerű validálása

A megszorításokat hozzáadhatjuk a táblához már létrehozáskor (CREATE), vagy később is a tábla módosításával (ALTER). Utóbbi esetén a már beszúrt adatok okozhatnak problémát

Tartalom

27 Megszorítások

- Elsődleges kulcs
- Idegen kulcs
- Egyedi értékek UNIQUE

Elsődleges kulcs

- Az elsődleges kulcs egyértelműen meghatároz egy rekordot (sort) a táblában.
- Az elsődleges kulcs állhat több mezőből, ekkor **összetett kulcsról** beszélünk.
- Egy táblának csak egy elsődleges kulcsa lehet!
- Kitöltése kötelező! Amennyiben nem adjuk meg, hogy legyen NOT NULL ezt az adatbázis kezelő hozzáteszi a háttérben.

<https://dev.mysql.com/doc/refman/8.0/en/create-table.html>

Elsődleges kulcs megadása

- Elsődleges kulcs megadás a mező tulajdonságként:

MySQL

```
CREATE TABLE `pelda` (  
  `id` INT NOT NULL PRIMARY KEY  
);
```

- Elsődleges kulcs megadás a mezők felsorolása után:

MySQL

```
CREATE TABLE `pelda` (  
  `id` INT NOT NULL,  
  PRIMARY KEY (`id`)  
);
```

A fenti két utasítás ugyanazt eredményezi!

Összetett (elsődleges) kulcs hibás példa

Az elsődleges kulcs állhat több mezőből, ekkor **összetett kulcs**ról beszélünk.

```
CREATE TABLE `pelda` (  
  `nev` VARCHAR(20) NOT NULL PRIMARY KEY,  
  `kor` INT(11) NOT NULL PRIMARY KEY,  
  `cim` VARCHAR(20)  
);
```

Szintaktikai hiba

Közvetlen tulajdonságként nem tudjuk megadni.

Többszörös elsődleges kulcs definiálás.

Hibakód: #1068

Összetett (elsődleges) kulcs helyes példa

A mezők felsorolása után adhatjuk meg

MySQL

```
CREATE TABLE `pelda` (  
  `nev` VARCHAR(20) NOT NULL,  
  `kor` INT(11) NOT NULL,  
  `cim` VARCHAR(20),  
  PRIMARY KEY (`nev` , `kor`)  
);
```

Tartalom

27 Megszorítások

- Elsődleges kulcs
- Idegen kulcs
- Egyedi értékek UNIQUE

Hivatkozási integritás

- A normalizálás során az adatok több táblába kerülhetnek.
- A hivatkozási integritás akkor áll fent, ha igaz, hogy ha az egyik táblából hivatkozunk egy másik táblára, akkor ez csak létező elemre történhessen.
- A hivatkozási integritást az idegen kulcsok biztosítják.
- A megszorítás következményei
 - Új rekord hozzáadása egy kapcsolt táblához csak akkor lehetséges, ha az elsődleges táblában már létezik egy ugyanolyan rekord.
 - Az elsődleges tábla elsődleges kulcsának értékét nem módosíthatjuk, amíg a kapcsolt táblában (másodlagos tábla) tartoznak hozzá adatok.
 - Nem lehet az elsődleges táblából olyan rekordot törölni, amelyhez tartoznak rekordok a másodlagos, kapcsolódó táblában

Linkek:

- http://www.nemesvamosiskola.hu/dokumentumok/m5/m5_elmelet.pdf
- https://en.wikipedia.org/wiki/Referential_integrity

FOREIGN KEY

Definition

Az olyan attribútum(ok)at, amelyek egy másik relációban alkotnak kulcsot idegen kulcsnak nevezzük.

```
[CONSTRAINT [symbol]] FOREIGN KEY  
[index_name] (col_name, ...)  
REFERENCES tbl_name (col_name,...)  
[ON DELETE reference_option]  
[ON UPDATE reference_option]
```

reference_option:

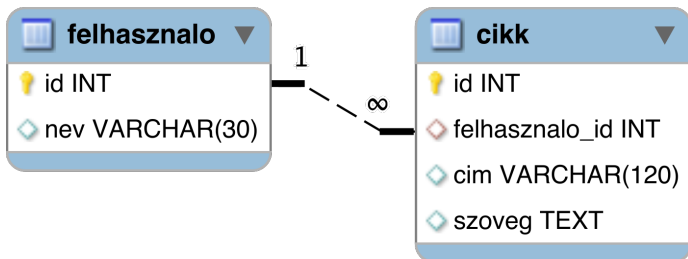
RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

<https://dev.mysql.com/doc/refman/8.0/en/create-table-foreign-keys.html>

Idegenkulcs opciói

- RESTRICT** Nem engedi a törlést/módosítást (ez az alapértelmezett)
- NO ACTION** MySQL-ben megegyezik a RESTRICT hatásával (SQL szabvány)
- CASCADE** A szülő táblában történő törlést/módosítást végrehajtja a gyerek táblán is
- SET NULL** A szülő táblában történő törlés/módosítás esetén a gyerek táblában NULL-ra állítja az értéket. Nem lehet NOT NULL az érték, mert az gondot okozna!
- SET DEFAULT** InnoDB és az NDB is figyelmen kívül hagyja (van, de minek)

Idegenkulcs példa (blog és szerző) I.



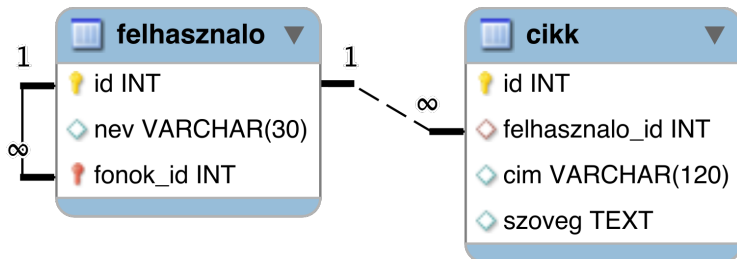
Idegenkulcs példa (blog és szerző) II.

MySQL

```
ALTER TABLE `cikk`  
  ADD constraint `fk_cikk_felhasznalo_id`  
    FOREIGN KEY (`felhasznalo_id`)  
    REFERENCES `felhasznalo` (`id`)  
    ON DELETE CASCADE  
    ON UPDATE CASCADE;
```

- A felhasználó törlése esetén a bejegyzés is törlésre kerül

Idegenkulcs példa (szerző és főnöke) I.



Idegenkulcs példa (szerző és főnöke) II.

MySQL

```
ALTER TABLE `felhasznalo`  
  ADD COLUMN `fonok_id` INT,  
  ADD CONSTRAINT `fk_felhasznalo_fonok_id`  
    FOREIGN KEY (`fonok_id`)  
    REFERENCES `felhasznalo` (`id`)  
    ON DELETE RESTRICT  
    ON UPDATE RESTRICT;
```

- A főnök törlését nem engedni, amíg van hozzá tartozó beosztott dolgozó

Idegenkulcs törlése

- Az idegen kulcs törlésénél jól jön, ha tudjuk hogy milyen nevet adtunk neki!

MS SQL, ORACLE

```
ALTER TABLE felhasználó  
DROP CONSTRAINT fk_fonok_id;
```

MySQL

```
ALTER TABLE felhasználó  
DROP FOREIGN KEY fk_fonok_id;
```

Idegen kulcs által okozott hibák

Error Code: 1451 Cannot delete or update a parent row: a foreign key constraint fails.

(A megadott sor(ok) nem törölhető(ek) idegen kulcs miatt)

Error Code: 1217 Cannot delete or update a parent row: a foreign key constraint fails

(A tábla nem törölhető idegen kulcs miatt)

Megoldás:

- Ellenőrzés kikapcsolása: **SET** foreign_key_checks = 0;
- Ellenőrzés visszakapcsolása: **SET** foreign_key_checks = 1;

Tartalom

- 27 Megszorítások
 - Elsődleges kulcs
 - Idegen kulcs
 - Egyedi értékek UNIQUE

Egyedi mező (UNIQUE)

MySQL

```
ALTER TABLE `felhasznalo`  
  ADD UNIQUE(`email`);
```

- Az adott oszlopban ugyanaz az érték kétszer nem szerepelhet!
- Több mező megadásakor csak akkor számít azonosnak két sor, ha mindegyik felsorolt mező megegyezik.

Tartalom I

- 28 Index
 - 0NF
 - 1NF

Az indexek típusai

PRIMARY KEY Elsődleges kulcs

UNIQUE Egyedi értékek

INDEX Indexelés rendezéshez, kereséshez

FULLTEXT Szöveg vagy szövegrészlet kereséséhez

INDEX általános leírása

- Az elsődleges kulcs létrehozásakor létrejön hozzá az index is
- Az idegen kulcs létrehozásakor létrejön hozzá az index is

Index létrehozása:

```
CREATE INDEX `index_neve`  
    ON `tabla_neve` (mezo1, mezo2, ...);
```

Index eldobása

```
DROP INDEX `index_neve`  
    ON `tabla_neve`;
```

Egyedi mező (UNIQUE)

- Az adott oszlopban ugyanaz az érték kétszer nem szerepelhet!

MySQL

```
ALTER TABLE `felhasznalo`  
  ADD UNIQUE(`email`);
```

- Több mező megadásakor csak akkor számít azonosnak két sor, ha mindegyik felsorolt mező megegyezik.

INDEX

Index létrehozása 1 mezőre:

```
CREATE INDEX IDX_1 ON auto (gyarto);
```

MySQL

Index eldobása:

```
DROP INDEX IDX_1 ON auto;
```

MySQL

Az indexek előnyei és hátrányai

- Előnyei

- Amikor adatokat kérünk le (SELECT) akkor gyorsít a lekérdezés eredményén.
- A WHERE feltételnek megfelelő sorokat hamarabb megtalálja.
- Gyorsítja a MIN() és a MAX() függvényeket
- Gyorsítja a csoportosítást (GROUP BY)

- Hátrányai

- Cserébe az adatok beszúrását lassítja.

Tartalom

- 28 Index
 - 0NF
 - 1NF

0. normálforma (0NF)

- Tartalmaz többértékű mezőt.
- Így jelöljük, hogy semelyiknek sem felel meg.
- Alternatív jelölés: UNF (Unnormalized Form)

Tartalom

- 28 Index
 - 0NF
 - 1NF

1. normálforma (1NF)

Definition (1. normálforma)

- Minden rekord különbözik
- Rekordonként megegyezik a mezők száma és sorrendje
- Nincsenek benne többértékű mezők!
- Jellemző rá a redundancia

1NF példa: Ember

Vegyük az *Ember* (név, kor, hajszín) relációt

| név | kor | hajszín |
|--------------|-----|--------------|
| Pap Barnabás | 28 | barna |
| Szőke Éva | 46 | vörös, szőke |
| Bíró Péter | 55 | fekete |

- A hajszín attribútum lehet **többértékű**
- **Nem felel meg az első normálformának**

1NF példa: Ember

Bővítsük a relációt, újabb attribútummal, hogy elkerüljük többértékű mezőt: *Ember* (név, kor, hajszín1, hajszín2)

| név | kor | hajszín1 | hajszín2 |
|--------------|-----|----------|----------|
| Pap Barnabás | 28 | barna | NULL |
| Szőke Éva | 46 | vörös | szőke |
| Bíró Péter | 55 | fekete | NULL |

- Mi van azokkal, akiknek nincs második szín megadva?
 - Rengeteg NULL értéket eredményez
- Mi van akkor, ha valakinek három színű a haja?
 - Nem tudjuk eltárolni az adatbázisba, ha 1NF-nak meg szeretnénk felelni
- **Nem túl jó megoldás**

1NF példa: Ember

Maradjunk az eredeti *Ember* (név, kor, hajszín) relációnál

| név | kor | hajszín |
|--------------|-----|---------|
| Pap Barnabás | 28 | barna |
| Szőke Éva | 46 | vörös |
| Szőke Éva | 46 | szőke |
| Bíró Péter | 55 | fekete |

- A többértékű mezőket tüntessük fel külön sorokban
- Szőke Éva kétszer is (azaz redundánsan) szerepel a két hajszín miatt
- Nincs fölösleges kitöltetlen (NULL) érték
- Eltárolható az adatbázisban tetszőleges számú hajszín
- Teljesült az 1NF

Tartalom I

29 Kapcsolatok típusai

Kapcsolat

A relációkat a számosságuk alapján az alábbi típusokba soroljuk:

- Egy az egyhez (1:1)
- Egy a többhöz (1:N)
- Több a többhöz (N:M)

Egy az egyhez (1:1)

Egy reláció soraihoz legfeljebb egy sor tartozik a másik relációban.

- Házasság (monogám kapcsolatban)
- Elnök \iff Ország
 - Egy országnak egy elnöke van, és senki se lehet több ország elnöke egyszerre

Egy a többhöz (1:N)

Egy reláció soraihoz akár több sor is tartozhat a másik relációban.

- Anyuka \longleftrightarrow gyerek vagy Apuka \longleftrightarrow gyerek
 - Egy anyának több gyereke is lehet.
 - Egy apának több gyereke is lehet.
 - A szülő gyerek kapcsolat már nem tartozik ide!
- Gyártó \longleftrightarrow Autó
 - Egy gyártó több típust is gyárthat. Az is lehet, hogy csak egyet, de ez ritka.
 - Amennyiben egy autót egy másik gyártó is gyárt, azt a saját nevén és minimális módosításokkal teszi, így a két autó nem ugyanaz!
- poligínia (többnejűség)
 - Egy férjnek lehet több felesége
- poliandria (többférjűség)
 - Egy feleségnek lehet több férje (poliandria, többférjűség)

Több a többhöz (N:M)

Egy reláció soraihoz akár több sor is tartozhat a másik relációban és ez visszafelé is teljesül.

- Film \longleftrightarrow Színész
 - Egy filmben több színész is szerepel, de egy színész általában több filmben is játszik.
 - Lehet olyan film, amiben csak egy színész szerepel
 - Lehet olyan szereplő, aki csak egy filmben szerepel
 - Lehet olyan film, ahol nincs szereplő (pl.: természetfilm)
 - Lehet olyan színész, aki (még) egyetlen filmben sem szerepelt
 - **A legtágabb értelemben vizsgáljuk a kapcsolatot**
- Szülő \longleftrightarrow Gyerek
 - Egy gyereknek több szülője is van (1 apuka, 1 anyuka). Egy szülőnek lehet egy vagy több gyereke, de az is elhet, hogy nincs neki!

Tartalom I

- 30 2NF - A második normálforma
 - Elsődleges és leíró attribútumok
 - Részleges funkcionális függőség
 - Teljes funkcionális függőség
 - 2NF definíciója
 - 2NF-ra alakítás

Tartalom

- 30 2NF - A második normálforma
 - Elsődleges és leíró attribútumok
 - Részleges funkcionális függőség
 - Teljes funkcionális függőség
 - 2NF definíciója
 - 2NF-ra alakítás

Elsődleges és másodlagos/leíró attribútum

Definition (Elsődleges attribútum)

Azokat az attribútumokat, melyek részei a reláció bármelyik kulcs(jelölt)jének elsődleges attribútumnak nevezzük

Definition (Másodlagos (vagy leíró) attribútum)

Azokat az attribútumokat, melyek nem része egyetlen kulcs(jelölt)nek sem másodlagos, vagy más szóval leíró attribútumnak nevezzük

Figyelem!

Az **elsődleges attribútum** nem összekeverendő az **elsődleges kulccsal**!

Elsődleges és másodlagos/leíró attribútum

- Vegyük az alábbi Személy relációt:
 - Személy (szem_szám, név, anyja_neve, szül_hely, szül_idő, cím, tel)
- Nem triviális funkcionális függőségek:
 - {név, szül_idő, szül_hely, anyja_neve} \rightarrow {cím, tel}
 - {szem_szá
- Kulcs(jelölt)ek:
 - {név, szül_idő, szül_hely, anyja_neve}
 - {szem_szá
- Elsődleges kulcs:
 - {szem_szá
- Elsődleges attribútumok:
 - {szem_szá
- Másodlagos/leíró attribútumok
 - {cím, tel}

Tartalom

30 2NF - A második normálforma

- Elsődleges és leíró attribútumok
- Részleges funkcionális függőség
- Teljes funkcionális függőség
- 2NF definíciója
- 2NF-ra alakítás

Részleges funkcionális függőség

$$R(A_1, A_2, \dots, A_n, \dots, B_1, B_2, \dots, B_n, C_1, C_2, \dots, C_n)$$

Definition

A C attribútumhalmaz funkcionálisan függ az A és B attribútumhalmaztól együtt, de külön az A attribútumhalmaztól vagy külön a B attribútumhalmaztól is függ a C attribútumhalmaz.

- $(A + B \rightarrow C, \text{létezik, hogy } A \rightarrow C \text{ vagy } B \rightarrow C)$

Részleges funkcionális függőség

Vegyük a következő relációt:

Könyvtár(isbn, cím, hossz, olvasójegy, elvitte, visszahozta)

- Nem triviális funkcionális függőségek:

- $\{ isbn \} \rightarrow \{ cím, hossz \}$
- $\{ isbn, olvasójegy, elvitte \} \rightarrow \{ visszahozta \}$ (visszahozta lehet kitöltetlen)
- $\{ isbn, olvasójegy, elvitte \} \rightarrow \{ cím, hossz, visszahozta \}$

- A reláció minden attribútuma szerepel a függőség valamelyik oldalán

- **Szuperkulcs:** $\{ isbn, olvasójegy, elvitte \}$
- **Kulcs(jelölt):** $\{ isbn, olvasójegy, elvitte \}$
- **Elsődleges kulcs:** $\{ isbn, olvasójegy, elvitte \}$
- **Elsődleges attribútumok halmaza:** $\{ isbn, olvasójegy, elvitte \}$
- **Leíró attribútumok halmaza:** $\{ cím, hossz, visszahozta \}$
- A könyv **címe** és **hossza** csak az **ISBN** attribútumtól függ, ami a kulcs egy része, így ez egy **részleges funkcionális függőség**

Tartalom

30 2NF - A második normálforma

- Elsődleges és leíró attribútumok
- Részleges funkcionális függőség
- **Teljes funkcionális függőség**
- 2NF definíciója
- 2NF-ra alakítás

Teljes funkcionális függőség

$$R(A_1, A_2, \dots, A_n, \dots, B_1, B_2, \dots, B_n, C_1, C_2, \dots, C_n)$$

Definition

A C attribútumhalmaz funkcionálisan függ az A és B attribútumhalmaztól együtt, de külön külön nem.

- $(A + B \rightarrow C, \text{ de sem } A \rightarrow C \text{ sem } B \rightarrow C)$

Másképp megfogalmazva: **Nem** létezik olyan attribútum, ami a kulcs egy részétől függ, nem a teljes egésztől

Teljes funkcionális függőség

Vegyük a következő relációt:

Személy(név, szül_idő, szül_hely, anyja_neve, cím, tel)

- Nem triviális funkcionális függőségek:
 - $\{\text{név, szül_idő, szül_hely, anyja_neve}\} \rightarrow \{\text{cím, tel}\}$
- **Szuperkulcs:** $\{\text{név, szül_idő, szül_hely, anyja_neve}\}$
- **Kulcs(jelölt):** $\{\text{név, szül_idő, szül_hely, anyja_neve}\}$
- **Elsődleges kulcs:** $\{\text{név, szül_idő, szül_hely, anyja_neve}\}$
- **Elsődleges attribútumok halmaza:**
 $\{\text{név, szül_idő, szül_hely, anyja_neve}\}$
- **Leíró attribútumok halmaza:** $\{\text{cím, tel}\}$
- Nem létezik olyan leíró attribútum, ami függene bármelyik (itt egyetlen) kulcsjelölt egy részétől, így **teljes funkcionális függőség** áll fenn

Teljes funkcionális függőség

Vegyük a következő relációt:

Személy2 (név, személyi_szám, szül_idő, szül_hely, anyja_neve, cim, tel)

- **Nem triviális funkcionális függőségek:**

- $\{\text{személyi_szám}\} \rightarrow \{\text{név, szül_idő, szül_hely, anyja_neve, cim, tel}\}$
- $\{\text{név, szül_idő, szül_hely, anyja_neve}\} \rightarrow \{\text{cim, tel}\}$

- **Kulcs(jelölt)ek:**

- $\{\text{név, szül_idő, szül_hely, anyja_neve}\}$
- $\{\text{személyi_szám}\}$

- **Elsődleges kulcs:** $\{\text{személyi_szám}\}$

- **Elsődleges attribútumok halmaza:**

$\{\text{személyi_szám, név, szül_idő, szül_hely, anyja_neve}\}$

- **Leíró attribútumok halmaza:** $\{\text{cim, tel}\}$

- Nem létezik olyan leíró attribútum, ami függene **bármelyik** kulcs(jelölt) egy részétől, így **teljes funkcionális függőség** áll fenn

Tartalom

30 2NF - A második normálforma

- Elsődleges és leíró attribútumok
- Részleges funkcionális függőség
- Teljes funkcionális függőség
- 2NF definíciója
- 2NF-ra alakítás

2. normálforma (2NF)

Definition (2. norálforma)

- A reláció első normálformában van.
- A reláció minden nem elsődleges (leíró) attribútuma teljes funkcionális függőségben van az összes reláció kulccsal
- Nem lehet funkcionális függőség bármely kulcs(jelölt) egy részétől
- Ha minden kulcs(jelölt) egyszerű, azaz egy attribútumból áll, akkor 2NF teljesül
 - "Ha egy részből áll nem függhet semmi annak a részétől"
- Amennyiben nincs leíró attribútum, úgy a 2NF teljesül
 - "Ha nincs ami függjön, akkor ott nem is lehet függés"

Tartalom

30 2NF - A második normálforma

- Elsődleges és leíró attribútumok
- Részleges funkcionális függőség
- Teljes funkcionális függőség
- 2NF definíciója
- 2NF-ra alakítás

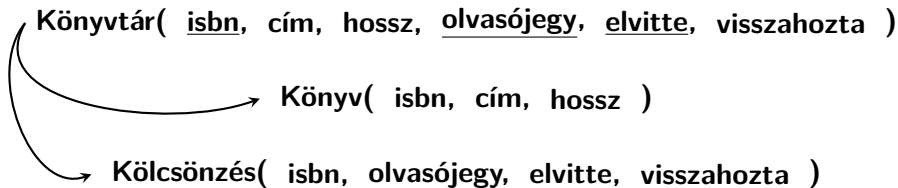
2. normálforma (2NF) példa

Vegyük a következő relációt:

Könyvtár(isbn, cím, hossz, olvasójegy, elvitte, visszahozta)

- Nem triviális funkcionális függőségek:
 - $\{ isbn \} \rightarrow \{ cím, hossz \}$
 - $\{ isbn, olvasójegy, elvitte \} \rightarrow \{ visszahozta \}$
 - $\{ isbn, olvasójegy, elvitte \} \rightarrow \{ cím, hossz, visszahozta \}$
- **Kulcs(jelölt):** $\{ isbn, olvasójegy, elvitte \}$
- **Elsődleges kulcs:** $\{ isbn, olvasójegy, elvitte \}$
- **Elsődleges attribútumok halmaza:** $\{ isbn, olvasójegy, elvitte \}$
- **Leíró attribútumok halmaza:** $\{ cím, hossz, visszahozta \}$
- A könyv **címe** és **hossza** csak az **ISBN** attribútumtól függ, ami a kulcs egy része, így ez egy **részleges funkcionális függőség**

2. normálforma (2NF) példa - dekompozíció



- Megoldás: **dekompozíció**, azaz bontsuk szét a relációt több kisebbre a funkcionális függőségek alapján
- A $\{isbn\} \rightarrow \{cím, hossz\}$ függőség alapján jött létre a **Könyv** tábla
- A reláció "bal oldala" és a többi attribútum lesz a másik táblánk.

2. normálforma (2NF) példa - dekompozíció

Állapítsuk meg a két új reláció kulcsát

- *Könyv* (isbn, cím, hossz)
 - **Nem triviális funkcionális függőség:** $\{\text{isbn}\} \rightarrow \{\text{cím}, \text{hossz}\}$
 - **Szuperkulcs:** $\{\text{isbn}\}$
 - **Kulcs(jelölt):** $\{\text{isbn}\}$
 - **Elsődleges kulcs:** $\{\text{isbn}\}$
- *Kölcsönzés* (isbn, olvasójegy, elvitte, visszahozta,)
 - **Nem triviális funkcionális függőség:**
 $\{\text{isbn}, \text{olvasójegy}, \text{elvitte}\} \rightarrow \{\text{visszahozta}\}$
 - **Szuperkulcs:** $\{\text{isbn}, \text{olvasójegy}, \text{elvitte}\}$
 - **Kulcs(jelölt):** $\{\text{isbn}, \text{olvasójegy}, \text{elvitte}\}$
 - **Elsődleges kulcs:** $\{\text{isbn}, \text{olvasójegy}, \text{elvitte}\}$

2. normálforma (2NF) példa

Ellenőrizzük, hogy visszakapható -e az eredeti reláció

Könyv (isbn, cím, hossz)

Kölcsönzés (isbn, olvasójegy, elvitte, visszahozta,)

- Egy könyv többször is kikölcsönözhető, így 1:N a kapcsolat a Könyv és a Kölcsönzés relációk között
- A Könyv reláció kulcsának szerepelnie kell a Kölcsönzés relációban
- Ellenőrizzük, hogy megfelel-e a 2NF-nek a két új reláció, ha igen készen is vagyunk

Tartalom I

- 31 3NF - A harmadik normálforma
 - Tranzitív funkcionális függőség
 - 3NF definíciója
 - 3NF-ra alakítás

Tartalom

- 31 3NF - A harmadik normálforma
 - Tranzitív funkcionális függőség
 - 3NF definíciója
 - 3NF-ra alakítás

Tranzitív függőség

$$R(A_1, A_2, \dots, A_n, \dots, B_1, B_2, \dots, B_n, C_1, C_2, \dots, C_n)$$

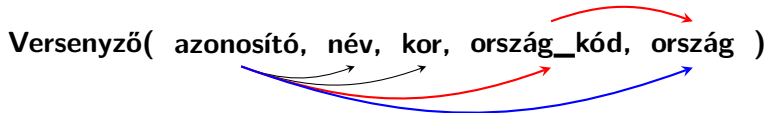
Definition (Tranzitív függőség)

A C attribútumok funkcionálisan függenek az A attribútumoktól, de létezik olyan B attribútum ami függ az A attribútumhalmaztól, és a C attribútumhalmaz függ a B attribútumhalmaztól.

- $A \rightarrow C$, de létezik B , hogy $A \rightarrow B$ és $B \rightarrow C$

Tranzitív funkcionális függőség példa

Vegyük a következő relációt:



- Nem triviális funkcionális függőségek:
 - $\{\text{azonosító}\} \rightarrow \{\text{név, kor, ország_kód, ország}\}$, ami magába foglalja:
 - $\{\text{azonosító}\} \rightarrow \{\text{név}\}$
 - $\{\text{azonosító}\} \rightarrow \{\text{kor}\}$
 - $\{\text{azonosító}\} \rightarrow \{\text{ország_kód}\}$
 - $\{\text{azonosító}\} \rightarrow \{\text{ország}\}$
 - $\{\text{ország_kód}\} \rightarrow \{\text{ország}\}$
- Az *azonosító* attribútum közvetlen meghatározza az ország attribútumot
- Ugyanakkor az *azonosító* attribútum az *ország_kód*-on keresztül közvetetten is meghatározza az ország attribútumot
- Utóbbi lesz a **tranzitív funkcionális függőség**

Tartalom

- 31 3NF - A harmadik normálforma
 - Transzitiv funkcionális függőség
 - 3NF definíciója
 - 3NF-ra alakítás

3. normálforma (3NF)

Definitions (3. normál forma)

- A reláció második normál formában van.
- A reláció nem tartalmaz funkcionális függőséget a nem elsődleges attribútumok között.
- Nincs benne tranzitív függőség
- Ha nincs másodlagos/leíró attribútum, akkor 3NF teljesült.

Tartalom

- 31 3NF - A harmadik normálforma
 - Transzitiv funkcionális függőség
 - 3NF definíciója
 - 3NF-ra alakítás

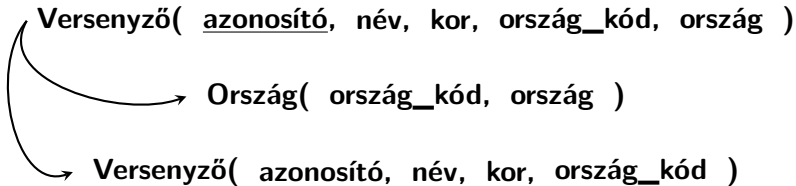
3. normálforma (3NF) példa

Vegyük a következő relációt:



- Nem triviális funkcionális függőségek:
 - $\{\text{azonosító}\} \rightarrow \{\text{név}, \text{kor}, \text{ország_kód}, \text{ország}\}$
 - $\{\text{ország_kód}\} \rightarrow \{\text{ország}\}$
- **Kulcs(jelölt):** $\{\text{azonosító}\}$
- **Elsődleges kulcs:** $\{\text{azonosító}\}$
- **Elsődleges attribútumok halmaza:** $\{\text{azonosító}\}$
- **Leíró attribútumok halmaza:** $\{\text{név}, \text{kor}, \text{ország_kód}, \text{ország}\}$
- A $\{\text{ország_kód}\} \rightarrow \{\text{ország}\}$ funkcionális függőség sérti 3NF-t, ami tiltja a leíró attribútumok közti funkcionális függőséget

3. normálforma (3NF) példa



- Megoldás: **dekompozíció**, azaz bontsuk szét a relációt több kisebbre a funkcionális függőségek alapján
- Az $\{\text{ország_kód}\} \rightarrow \{\text{ország}\}$ függőség alapján jön létre az **Ország** reláció
- A reláció "bal oldala" és a többi attribútum lesz a másik relációnk.

3. normálforma (3NF) példa - dekompozíció

Állapítsuk meg a két új relációk kulcsait

- *Versenyző* (azonosító, név, kor, ország_kód)
 - **Nem triviális funkcionális függőség:**
 $\{\text{azonosító}\} \rightarrow \{\text{név, kor, ország_kód}\}$
 - **Szuperkulcs:** $\{\text{azonosító}\}$
 - **Kulcs(jelölt):** $\{\text{azonosító}\}$
 - **Elsődleges kulcs:** $\{\text{azonosító}\}$
- *Ország* (ország_kód, ország,)
 - **Nem triviális funkcionális függőség:** $\{\text{ország_kód}\} \rightarrow \{\text{ország}\}$
 - **Szuperkulcs:** $\{\text{ország_kód}\}$
 - **Kulcs(jelölt):** $\{\text{ország_kód}\}$
 - **Elsődleges kulcs:** $\{\text{ország_kód}\}$

3. normálforma (3NF) példa

Ellenőrizzük, hogy visszakapható -e az eredeti reláció

Versenyző (azonosító, név, kor, ország_kód)

Ország (ország_kód, ország)

- Egy országnak lehet több versenyzője is (ahol hivatalosan bejegyzett sportoló), így 1:N a kapcsolat az Ország és a Versenyző relációk között
- Az ország reláció kulcsának szerepelnie kell a versenyző attribútumai között
- Ellenőrizzük, hogy megfelel-e a 3NF-nek a két új reláció, ha igen készen is vagyunk

Tartalom I

32 Mi az az EK model?

Mi az az EK diagram?

- Az EK modell az egyed-kapcsolat modell rövidítése
- Angolul ER model (Entity Relationship Model)
- Az EK modell-t **EK diagramon** ábrázoljuk
- Angolul Entity Relationship Diagram, röviden ERD
- Az egyed-kapcsolat modellezést Peter Chen fejlesztette ki 1976-ban
- Sokféle jelölés rendszer létezik, Chen jelölésrendszere kerül bemutatásra

Tartalom I

33 EK diagram elemei

- Egyed (Entity)
- Attribútum (Attribute)
- Kapcsolat (Relation)

Tartalom

33 EK diagram elemei

- Egyed (Entity)
- Attribútum (Attribute)
- Kapcsolat (Relation)

Egyed (entity)

Egyed

- Az ER modell által kezelt alapvető objektum az **egyed**,
- amely a valós világnak egy olyan darabja, amely önálló léttel bír.
- Az egyed lehet fizikai szinten létező objektum
 - személy
 - autó
 - dolgozó
- vagy lehet fogalmi szinten létező objektum
 - vállalat
 - foglalkozás
 - tárgy.
- **Jelölés:** Téglalap
- Itt nem egy konkrét egyedet értünk alatta, hanem egyedek egy halmazát

Egyed (entity) példák

Autó

Ház

Tantárgy

Dolgozó

Tartalom

- 33 EK diagram elemei
 - Egyed (Entity)
 - **Attribútum (Attribute)**
 - Kapcsolat (Relation)

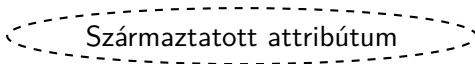
Attribútum (attribute)



Attribútum

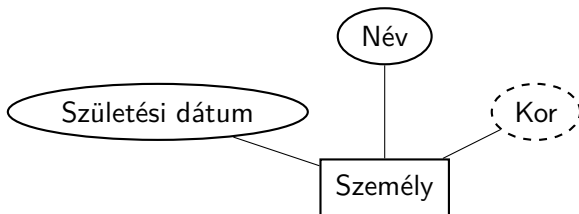
- Minden egyednek vannak attribútumai — az őt leíró tulajdonságok.
- Például egy **dolgozó** egyedet
 - nevével
 - életkorával
 - címével
 - fizetésével
 - és foglalkozásával lehet leírni.
- Egy konkrét egyed minden egyes attribútumához tartozik egy érték.
- Azokat az attribútumokat, amelyeket nem bontunk részekre, egyszerű vagy **atomi attribútumoknak** nevezzük.
- **Jelölés:** Ovális

Származtatott attribútum (derived attribute)



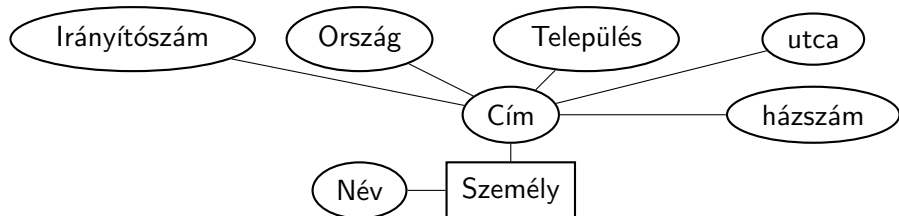
- Kiszámítható egy másik attribútumból
- Nem lesz eltárolva fizikailag az adatbázisban
- **Jelölés:** Szaggatott vonallal rajzolt ovális

Származtatott attribútum példa



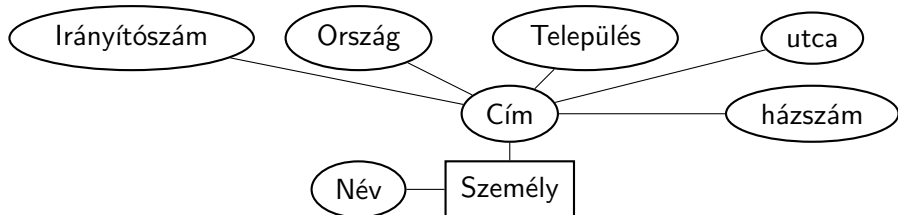
- A kor származtatott attribútum
- A születési dátum és az aktuális dátum ismeretében kikövetkeztethető

Összetett (composite) attribútum példa



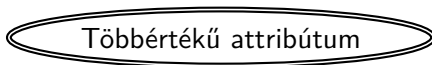
- A cím **összetett** attribútum
 - több részre bontható

Összetett (composite) attribútum példa



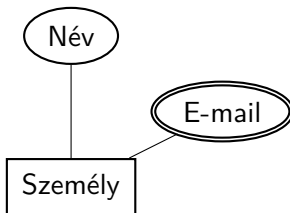
- A cím **összetett** attribútum
 - több részre bontható

Többértékű attribútum (multivalued attribute)



- **Jelölés:** dupla szegélyű ovális

Többértékű attribútum példa



- Az ábrán az E-mail többértékű attribútum
- Egy személyhez több emailcímet is szeretnénk eltárolni az adatbázisban
- Lehet olyan személy, akinek csak egy lesz, vagy annyi sem

Attribútumok csoportosítása

- Komplexitás
 - Egyszerű / Atomos
 - Összetett / Kompozit
- Számosság
 - Egyértékű
 - Többértékű
- Tárolás
 - Eltárolt
 - Származtatott

Tartalom

33 EK diagram elemei

- Egyed (Entity)
- Attribútum (Attribute)
- Kapcsolat (Relation)

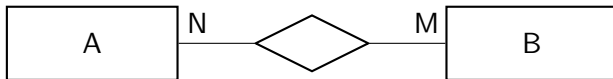
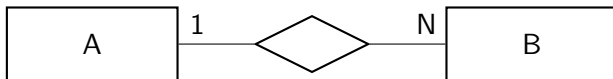
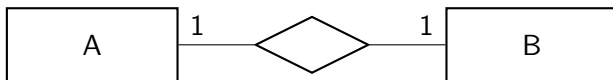
Kapcsolat (relation)



Kapcsolat: két vagy több egyed között határoz meg relációt

- **Jelölés:** rombusz

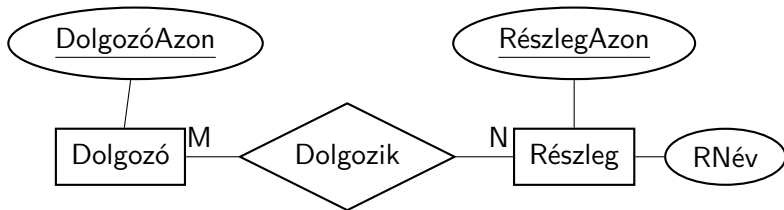
Kapcsolat példák



Tartalom I

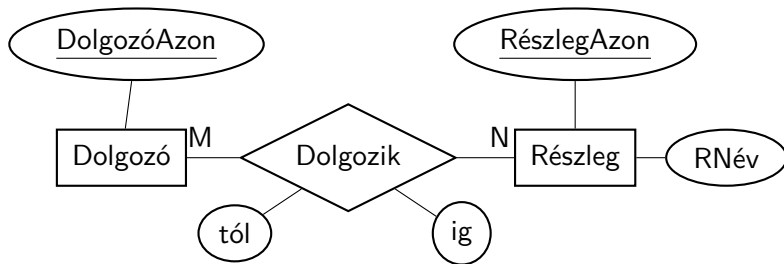
34 EK diagram példák

ER példa: Dolgozó és Részleg



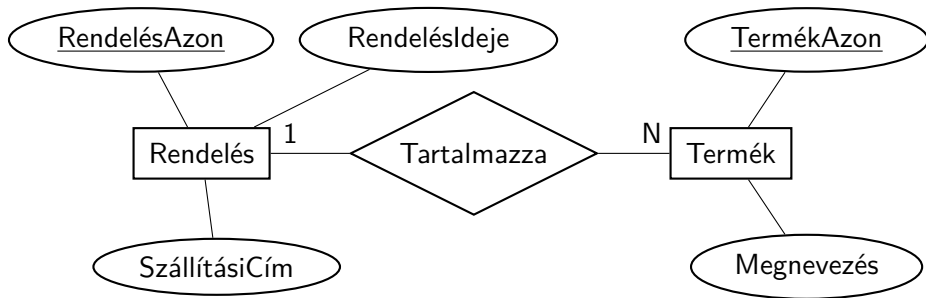
- Egy dolgozó több részlegen is dolgozhat
- Egy részlegen többen is dolgoznak
- A kapcsolat típusa: **több a többhöz** (M:N)
- A kulcs attribútumok aláhúzással lettek jelölve

ER példa: Dolgozó



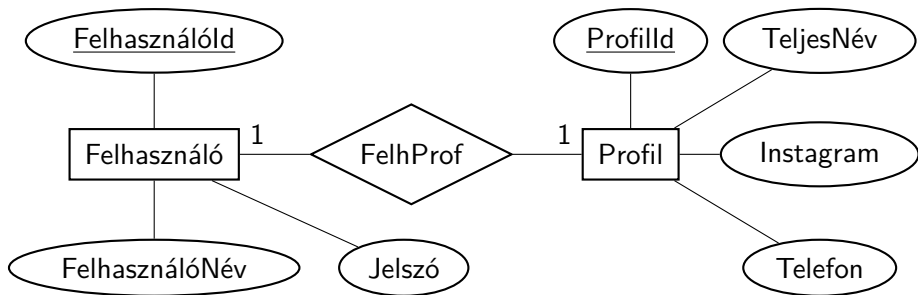
- A kapcsolatnak is lehetnek attribútumai
- Megadott időpontok között dolgozott az adott részlegen

ER példa: Rendelés



- Egy rendeléshez több termék is tartozhat
- Egy konkrét termék csak egy rendeléshez tartozik
 - Itt nem egy típusú telefonra kell gondolni, hanem egy konkrét telefonra, ami egyénileg beazonosítható
- A kapcsolat típusa: **egy a többhöz (1:N)**
- A kulcs attribútumok aláhúzással lettek jelölve

ER példa: Felhasználó és Profil



- Egy felhasználóhoz pontosan egy profil tartozik
- Egy profilhoz pontosan egy felhasználó tartozik
- A kapcsolat típusa: **egy az egyhez** (1:1)
- A kulcs attribútum aláhúzással lett jelölve
- A kapcsolat neve a két egyed nevének rövidítéséből áll össze

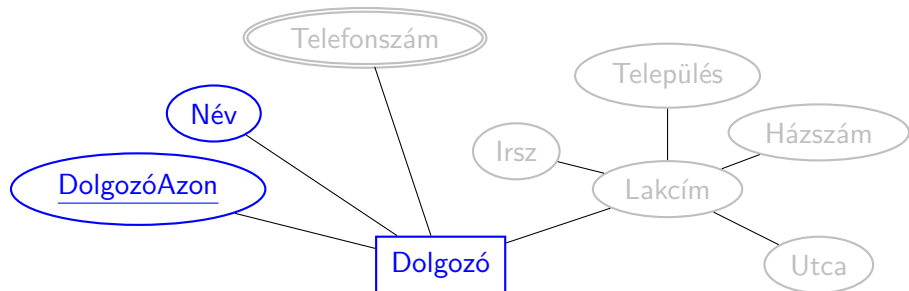
Tartalom I

- 35 ER modell leképezése
 - Egyedek leképezése
 - Bináris kapcsolat leképezése

Tartalom

- 35 ER modell leképezése
 - Egyedek leképezése
 - Bináris kapcsolat leképezése

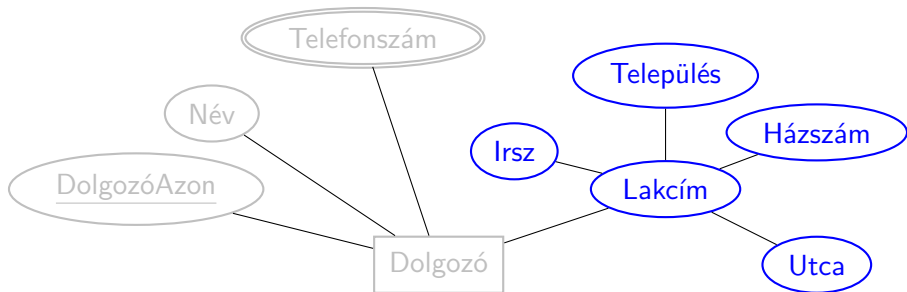
Egyedek és egyszerű attribútumok



Dolgozó(DolgozóAzon, Név)

- Az egyed egy relációnak felel meg.
- Az ER modell egyszerű attribútumai egy az egyben megfeleltethetők a reláció attribútumainak

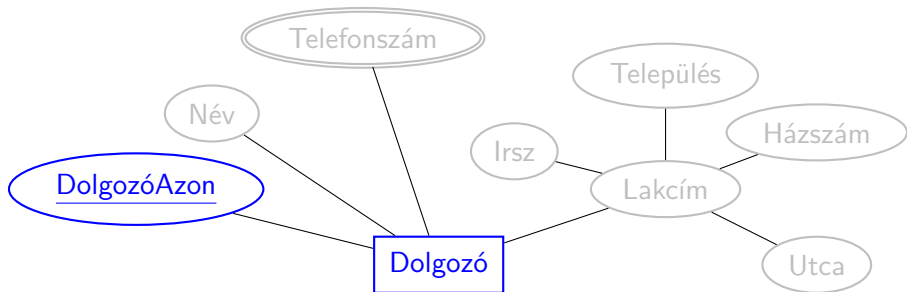
Összetett attribútumok



Dolgozó(DolgozóAzon, Név, Irsz, Település, Házzszám, Utca)

- Az összetett attribútum minden értéke egy-egy attribútum lesz a relációban
- A Lakcím attribútum **nem szerepel**, csak a részei

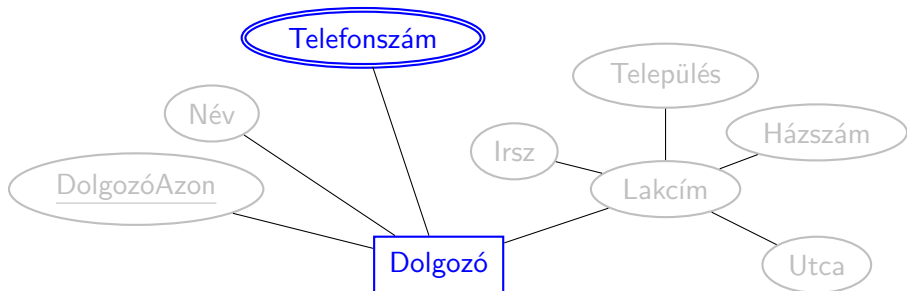
Kulcs



Dolgozó(DolgozóAzon, Név, Irsz, Település, Házzszám, Utca)

- Jelöljük a kulcsot is

Többértékű attribútum



Dolgozó(DolgozóAzon, Név, Irsz, Település, Házzszám, Utca)

Telefonkönyv(Telefonszám, DolgozóAzon)

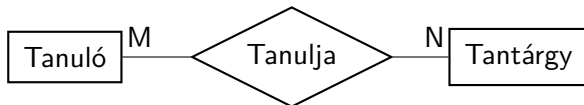
- A többértékű mező külön relációba kerül
- A Dolgozó reláció kulcsa idegenkulcsként jelenik meg az új relációban

Tartalom

- 35 ER modell leképezése
 - Egyedek leképezése
 - Bináris kapcsolat leképezése

Bináris kapcsolat

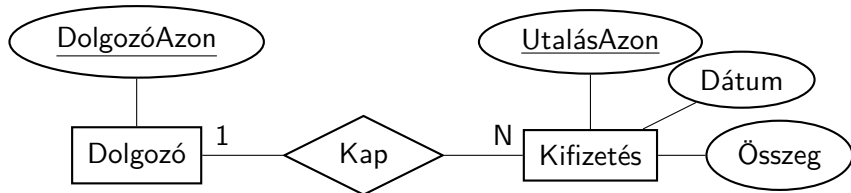
Egy **kapcsolat bináris**, amennyiben nem egy, nem három, hanem **két egyed között** áll fenn.



Típusai:

- 1:1
- 1:N
- M:N

Bináris kapcsolat (1:N és 1:1 esetében)

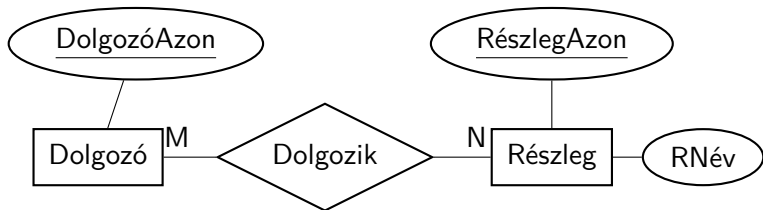


Dolgozó(DolgozóAzon, Név, Irsz, Település, Házszám, Utca)

Kifizetés(UtalásAzon, Dátum, Összeg, **DolgozóAzon**)

- A kifizetés egy reláció lesz
- A "Kap" kapcsolat nem kerül külön táblába
 - Az "1" oldalon álló egyed kulcsa lesz idegen kulcs az "N" oldalon álló relációban
- Az 1:1 kapcsolat esetén a megoldás hasonló, az egyik táblában feltüntetésre kerül a másik tábla elsődleges kulcsa

Bináris kapcsolat (M:N)



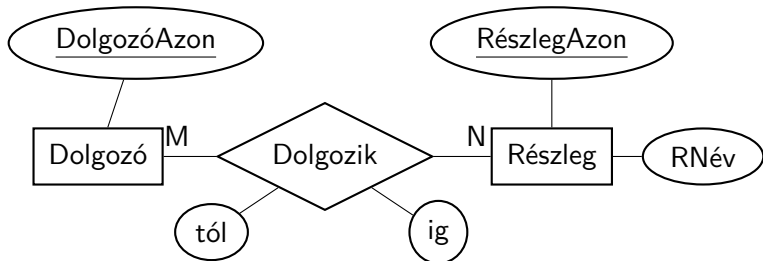
DOLGOZÓ(DolgozóAzon, Név, Irsz, Település, Utca, Házszám)

RÉSZLEG(RészlegAzon, RNév)

DOLGOZIK(DolgozóAzon, RészlegAzon)

- A Dolgozó és a Részleg egyedek egy az egyben leképezhetők.
- A kapcsolat itt egy új relációt hoz létre (Dolgozik)
 - Az összekötő táblában kell elhelyezni a két reláció kulcsát
- Öszetett kulcs lesz, ami egyben triviális is

Bináris kapcsolat (Több a többhöz - M:N)



DOLGOZÓ(DolgozóAzon, Név, Irsz, Település, Utca, Házzsám)

RÉSZLEG(RészlegAzon, RNév)

DOLGOZIK(DolgozóAzon, RészlegAzon, tól, ig)

- A kapcsolathoz tartozó attribútumok a Dolgozik relációban lesznek eltárolva
 - A "tól" is a kulcs része kell, hogy legyen