Adatbázis kezelés I.

Szövegfüggvények

Rostagni Csaba

2022.09.25

Ezen az órán... I

1. Szöveg függvények

Tartalom I

Tartalom

CONCAT()

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

- Összefűzi az argumentumként kapott értékeket

Linkek:

- MySQL dokumentáció: CONCAT()

CONCAT()

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

- Összefűzi az argumentumként kapott értékeket
- A számokat átalakítja szöveggé

Linkek:

- MySQL dokumentáció: CONCAT()

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

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

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

```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

CONCAT() példák

MySQL

```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

CONCAT() példák

MySQL

```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

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)

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 aliast a rendezésben

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 aliast a rendezésben
- A CONCAT-ben található rész szerint kell rendezni:
  `ROUND(`netto` * (1  + afa))`

CONCAT_WS()

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

- "Concatenate With Separator"

CONCAT_WS()

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

- "Concatenate With Separator"
- Összefűzi az argumentumként kapott értékeket

CONCAT_WS()

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

- "Concatenate With Separator"
- Összefűzi az argumentumként kapott értékeket
- Az első argumentum az elválasztó karakter

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

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

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

MySQL

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

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

MySQL

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

e
alma*barack*eper

Linkek:

- MySQL dokumentáció: CONCAT_WS()

Tartalom

UPPER()

```
UPPER(str)
```

- Nagybetűssé alakítja a szöveget (`str`)

UPPER()

```
UPPER(str)
```

- Nagybetűssé alakítja a szöveget (`str`)
- Alapértelmezetten a latin1 kódolást (cp1252 West European) használja

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 kezeli

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 kezeli
- Szinonímák erre a függvényre:

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 kezeli
- Szinonímák erre a függvényre:
  - `UCASE(str)`

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 kezeli
- Szinonímák erre a függvényre:
  - `UCASE(str)`

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 kezeli
- Szinonímák erre a függvényre:
  - `UCASE(str)`

MySQL

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

e
HELLO

LOWER()

```
LOWER(str)
```

- Kisbetűssé alakítja a szöveget (`str`)

LOWER()

```
LOWER(str)
```

- Kisbetűssé alakítja a szöveget (`str`)
- Alapértelmezetten a latin1 kódolást (cp1252 West European) használja

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 kezeli

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 kezeli
- Szinonímák erre a függvényre:

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 kezeli
- Szinonímák erre a függvényre:
  - `LCASE(str)`

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 kezeli
- Szinonímák erre a függvényre:
  - `LCASE(str)`

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 kezeli
- Szinonímák erre a függvényre:
  - `LCASE(str)`

MySQL

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

e
hello

Tartalom

LENGTH()

```
LENGTH(str)
```

- Megadja a szöveg (`str`), hosszát **byteokban**

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**

CHAR_LENGTH()

```
CHAR_LENGTH(str)
```

- Megadja a szöveg (`str`), hosszát **karakterekben**
- A multibyte karaktereket **egyszer** számolja

CHAR_LENGTH()

```
CHAR_LENGTH(str)
```

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

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

MySQL

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

e
3

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

MySQL

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

e
4

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

LENGTH() és CHAR_LENGTH() összehasonlítása

logikai hiba

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

e
30

LENGTH() és CHAR_LENGTH() összehasonlítása

logikai hiba

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

e
30

MySQL

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

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

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'` - ez az alapértelmezett), vagy
  - hogy tizedes vesszőt (`'hu_HU'`) használjon

MySQL

```sql
SELECT FORMAT(1234.123,2,'hu_HU') AS `formazott`
FROM DUAL;
```

formazott
1.234,12

Tartalom

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

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

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.

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

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:

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

MySQL

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

e
12

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

LOCATE() példák

MySQL

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

e
12

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

MySQL

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

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

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

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

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
Szekesfehervár

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

REPLACE() példák

MySQL

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

e
Szekesfehervár

- 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

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR len])
```

- Kivág egy részt a szövegből (`str`), a kezdő pozíciótól (`pos`) kezdve megadott számú (`len`) karaktert

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR len])
```

- Kivág egy részt a szövegből (`str`), a kezdő pozíciótól (`pos`) kezdve megadott számú (`len`) karaktert
- Az eredeti szöveget nem módosítja

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR 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:

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR 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()`

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR 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()

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

```
SUBSTRING(str [FROM pos] [FOR 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()`
- Hasonló függvények

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR 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()`
- Hasonló függvények
  - `LEFT()` az elejéről N karaktert vesz

SUBSTRING()

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

```
SUBSTRING(str [FROM pos] [FOR 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()`
- Hasonló függvények
  - `LEFT()` az elejéről N karaktert vesz
  - `RIGHT()` a végéről N karaktert vesz

SUBSTRING() példák

MySQL

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

e
fehérvár

- A `len` elhagyásával a szöveget a `pos`-tól a legvégéig veszi

SUBSTRING() példák

MySQL

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

e
fehérvár

- A `len` elhagyásával a szöveget a `pos`-tól a legvégéig veszi

MySQL

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

e
fehér

- A 7. karaktertől vesz 5 karaktert

SUBSTRING() példák

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

e
fehérvár

- A `len` elhagyásával a szöveget a `pos`-tól a legvégéig veszi

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

e
fehér

- A 7. karaktertől vesz 5 karaktert

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

e
vár

- Negatív `pos` esetén hátulrol lép vissza, majd a `len`-ben meghatározott karaktert veszi, annak elhagyásával a végéig

Tartalom

TRIM

```
TRIM([{BOTH | LEADING | TRAILING} [remstr] FROM] str)
```

- A `remstr` szöveget eltávolítja a `str` szövegből
  - A remstr elhagyásával a szóközöket veszi ki
- Levághat az elejéről `LEADING`, végéről `TRAILING` vagy mindkét oldalról `BOTH`
  - Ha nincs megadva, akkor alapértelmezetten mindkét oldalról leveszi
- `NULL`-t ad vissza, ha bármelyik paramétere NULL
- A több bájton tárolt akraktereket jól kezeli
- Az eredeti szöveget nem módosítja

TRIM examples

MySQL

```
SELECT␣TRIM('␣␣␣Hello␣World␣␣␣')␣FROM␣dual;
```

```
Hello World
```

- A szóközöket eltávolította mind a két oldlról

TRIM examples

MySQL

```
SELECT␣TRIM('␣␣␣Hello␣World␣␣␣')␣FROM␣dual;
```

```
Hello World
```

- A szóközöket eltávolította mind a két oldlról

MySQL

```
SELECT␣TRIM('*'␣FROM␣'***Hello*World***')␣FROM␣dual;
```

```
Hello*World
```

- A csillagokat eltávolította mind a két oldlról

TRIM examples

MySQL

```
SELECT␣TRIM('␣␣␣Hello␣World␣␣␣')␣FROM␣dual;
```

```
Hello World
```

- A szóközöket eltávolította mind a két oldlról

MySQL

```
SELECT␣TRIM('*'␣FROM␣'***Hello*World***')␣FROM␣dual;
```

```
Hello*World
```

- A csillagokat eltávolította mind a két oldlról

MySQL

```
SELECT␣TRIM(LEADING␣'*'␣FROM␣'***Hello*World***')␣FROM␣dual;
```

```
Hello*World***
```

- A csillagokat eltávolította, de csak az elejéről

Tartalom

```
REPEAT(str,n)
```

MySQL

```
SELECT REPEAT('Hello', 3) FROM dual;
```

```
HelloHelloHello
```

- A megadott szöveget ismétli N-szer

```
REPEAT(str,n)
```

MySQL

```
SELECT REPEAT('Hello', 3) FROM dual;
```

```
HelloHelloHello
```

- A megadott szöveget ismétli N-szer

```
LPAD(str,len,padstr)
```

MySQL

```
SELECT LPAD('7', 3, '0') FROM dual;
```

```
007
```

- A `str` szövegeg `len` hosszúvá bővíti a `padstr` szöveggel
- A kiegészítő karaktereket a szöveg elé rakja
- A `RPAD` a kiegészítő karaktereket a szöveg mögé teszi