Adatbázis kezelés I.

Típusok, adatbázis és tábla létrehozása/törlése

Rostagni Csaba

2023. február 5.

Ezen az órán... I

1. Típusok
2. Adatbázis létrehozása és törlése
3. Tábla létrehozása és törlése
4. Adatok beszúrása és törlése

Tartalom I

Tartalom

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_számok	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

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 65 535[1] között van.
- El kell az adat mellett azt is tárolni, hogy hány karakterből áll.
- Rövidebb szövegnél nem történik semmi.
- Túl hosszú szövegnél adatvesztés történik!

[1] Függ 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

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	429467295
BIGINT UNSIGNED	8	18446744073709551615

FLOAT(m,d)

Lebegőpontos szám tárolásására alkalmas típus.

m a karakterek száma

d a tizedesek száma

$$\text{FLOAT(7,4)} \rightarrow 777{,}1234$$

- Összesen 7 számjegy jelenik meg, ebből 3 az egész rész, 4 a törtrész
- Az M és N értének 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

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'közti időértékeket tud eltárolni.
- https://dev.mysql.com/doc/refman/5.5/en/time.html

Tartalom

BOOL/BOOLEAN

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

Tartalom I

Tartalom

Adatbázis létrehozása (minimális példa)

MySQL

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

- A legegyszerűbb parancs adatbázis létrehozásához.
- A karakterkódolás a szerver alapértelmezettbeállítása 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 (**C**ase **I**nsensitive)

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

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

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

Tartalom

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

Alapértelmezett értékek

- A "`DEFAULT ertek`" 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 határozunk meg értéket

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

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

Elsődleges kulcs

- Az elsődleges kulcs egyértelműen meghatároz egy rekordot (sort) a táblában.
- Az elsődleges kulcs állhat több mezőből, ekkor **összetett kulcs**ról beszélünk.
- Egy táblának csak egy elsődeleges kulcsa lehet!
- Kitöltése kötelező! Amenyiben 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.

Szintaktikai hiba

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

Közvetlen tulajdonságként nem tudjuk megadni.

Hibakód: #1068

Többszörös elsődleges kulcs definiálás.

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` CHAR(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

Táblák törlése

- Egy tábla és adatainak törlése:

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

- Egy tábla és adatainak törlése, ha létezik:

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

- Több tábla törlése az adatokkal együtt:

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

Tartalom I

Tartalom

Beszúrás minta

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

MySQL

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

HIBA 1241

Gyakori hiba a fölösleges, hibás zárójelezés!

Szintaktikai hiba

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

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

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.

MySQL

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

Figyelem!

Az adatok törlése végleges!

Tartalom

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 milló felett kereső dolgozót **véglegesen**