



# SQL alapok

---

- Típusok
- Séma definíció
- Adatbevitel



# Structured Query Language

---

- Szabvány– SQL-99
  - A legtöbb relációs DBMS támogatja (legalábbis kisebb eltérésekkel)
- Tömör, felhasználóközel
- Deklaratív
  - Azt mondjuk meg, hogy mit szeretnénk látni, a végrehajtás módját a DBMS-re bízuk
- „Angol”
- Case-insensitive
  - A szöveg literálok kivételével
  - A nyelvi elemeket nagy betűvel szokás



# SQL utasítások

---

## ➤ Adatleíró (**D**ata **D**efinition **L**anguage - DDL)

### ■ Felhasználói objektumok kezelése

- Új objektumok (adatbázisok, relációk, nézetek, kényszerek, ...) létrehozása
- Meglévő objektumok módosítása (pl reláció kibővítése), törlése
- CREATE, ALTER, DROP...

## ➤ Adatelérést vezérlő (**D**ata **C**ontrol **L**anguage – DCL)

### ■ Jogosultság szabályozás (GRANT)



# SQL utasítások – 2

---

- Adatmódosító (**D**ata **M**anipulation **L**anguage – DML)
  - Új adatok (példányok, entitások) beszúrása (INSERT)
  - Létező adatok módosítása (UPDATE)
  - Létező adatok törlése (DELETE)
  - Létező adatok lekérdezése, szűrése, rendezése (SELECT)
    - Data Query Language - DQL



# SQL típusok – Számok

- bit 0 .. 1
- tinyint 0 .. 255 , **MySQL: -127..128**
- smallint  $-2^{15} .. +2^{15}-1$
- int/integer  $-2^{31} .. +2^{31}-1$
- bigint  $-2^{63} .. +2^{63}-1$
- decimal[(p,s)]  $-10^{38} + 1 .. 10^{38} - 1$  (0 ≤ s ≤ p ≤ 38)
  - E.g. decimal(5,2): -999.99 .. 999.99
  - default: decimal(18,0)
- numeric ~ decimal
- float [(n)]  $-1.79 \cdot 10^{308} .. -2.23 \cdot 10^{-308}, 0, 2.23 \cdot 10^{-308} .. 1.79 \cdot 10^{308}$ 
  - n: bitek száma (1..53, default: 53)
- real ~float
- **MySQL: mindenhol alkalmazható unsigned változat is**



# SQL típusok – Sztringek

---

- **char(n)** – ASCII, fix hossz
- **varchar(n)** – ASCII, változó hossz  
=> Maximum: MSSQL: 8000, Oracle: 4000, MySql: 255
- **nchar(n), nvarchar(n)** – unicode  
=> Maximum: MSSQL: 4000, Oracle: 2000, MySql: 255
- **varchar(MAX), nvarchar(MAX)** – MSSQL,  $2^{31}-1$  ill  $2^{30}-1$  karakter
  - text, ntext (depricated)
- tinytext, text, mediumtext, longtext – MySQL
  - 255,  $2^{16}-1$ ,  $2^{24}-1$ ,  $2^{30}-1$  byte



# SQL típusok – Dátum és idő

---

- **date**
- **time**
- **datetime** (date+time)
- **timestamp** ~datetime, kisebb tartomány

**! Eltérő tartományok !**



# SQL típusok – Egyéb adat

---

## ➤ MySQL

- **binary(n), varbinary(n)** : maximum 255 byte
- tinyblob, **blob**, mediumblob, longblob
  - maximum 255,  $2^{16}-1$ ,  $2^{24}-1$ ,  $2^{32}-1$  byte
  - blob(m) => a megfelelő méretű típus választódik ki automatikusan





# SQL kiegészítések

---

- ; a sorok végén (MySQL-ben kell, MS SQL-ben opcionális)
- Comment: #, /\* \*/
  - Máshol: --
- Hibakezelés
  - Szintaktikai hibák
  - Végrehajtás közben történő hibák kezelése



# SQL – Adatbázis létrehozása

---

## ➤ Új adatbázis létrehozása

- CREATE DATABASE <adatbázisnév>

- PL **CREATE DATABASE** test

## ➤ Kapcsolódás meglévő adatbázishoz

- Ugyanazon usernek lehet több adatbázishoz is hozzáférése

- USE <adatbázisnév>

- PL **USE** test

- (ORACLE: nincs rá szükség, a készítő user neve (egyben az adatbázis neve) része a tábla nevének (pl Joe.Salary))



# SQL – Táblák (relációk) létrehozása

➤ **CREATE TABLE** <táblanév> (  
    <oszlop1> <típus1> [<megkötések1>],  
    <oszlop1> <típus2> [<megkötések1>],  
    ...,  
    [<tábla szintű megkötések>]  
)

```
CREATE TABLE Termek (  
    Id int,  
    Nev nvarchar(50),  
    Rakterkeszlet int )
```

```
CREATE TABLE Megrendeles(  
    Id int,  
    Megrendelo nvarchar(50),  
    Cim nvarchar(50),  
    Datum datetime);
```



# SQL – Virtuális oszlopok

---

- Származtatott érték

- Akár el is tárolódhat (hatékonysági okokból)

- **MySQL: egyelőre nem támogatott**

- MSSQL: Computed column

```
CREATE TABLE Termek (  
    Id int,  
    Nev nvarchar(50),  
    Ar int,  
    BruttoAr as (Ar*1.27)) VAGY as Ar*1.27 PERSISTED
```

- ORACLE: Virtual column

...

```
BruttoAr NUMBER(5,2) GENERATED ALWAYS as (Ar*1.27) VIRTUAL  
VAGY  
BruttoAr as Ar*1.27
```



# SQL – Új adat bevitele

---

➤ INSERT INTO <táblanév>

[(<oszlop1>, <oszlop2>, ...)]

VALUES(<érték1>, <érték2>, ...)

➤ PL:

```
INSERT INTO Termek (Id, Nev, Raktarkeszlet)
```

```
VALUES (1, 'Alma', 15)
```

■ Sorrend tetszőleges

```
INSERT INTO Termek VALUES (1, 'Alma', 15)
```

■ Az attribútumok sorrendjében



# SQL – NULL értékek

---

➤ Egy rekord egy attribútuma nincs beállítva, üres

■ Pl mert nem adunk neki értéket:

```
INSERT INTO Termek (Id, Raktarkeszlet) VALUES (1, 15)
```

■ Vagy mert NULL értéket adunk:

```
INSERT INTO Termek (Id, Nev, Raktarkeszlet) VALUES(1, NULL, 15)
```

➤ Nem csak stringre, bármilyen típusra lehetséges

■ INSERT INTO Termek VALUES (NULL, NULL, NULL)



# SQL – NULL értékek – 2

---

- Ha meg akarjuk tiltani, akkor használjuk a **NOT NULL** opciót

- ```
CREATE TABLE Termek (  
    Id int NOT NULL,  
    Nev nvarchar(50) NOT NULL,  
    Raktarkeszlet int NOT NULL  
)
```

- Hibát okoz pl.

```
INSERT INTO Termek (Id, Nev, Raktarkeszlet) VALUES (1, NULL, 15)
```



# SQL – Egyedi értékek, kulcsok

- Még nem megoldott az Id egyedisége, megengedett pl

```
INSERT INTO Termek VALUES (1, 'Alma', 15)
```

```
INSERT INTO Termek VALUES (1, 'Körte', 25)
```

- Megoldás: **unique, primary key**

- unique: különböző értékek, de megengedett a NULL

- primary key: különböző értékek, és a NULL nem megengedett

```
CREATE TABLE Termek (  
    Id int PRIMARY KEY,  
    Nev nvarchar(50) NOT NULL,  
    Raktarkeszlet int NOT NULL  
)
```

V

```
CREATE TABLE Termek (  
    Id int NOT NULL,  
    Nev nvarchar(50) NOT NULL,  
    Raktarkeszlet int NOT NULL,  
    PRIMARY KEY(Id)  
)
```





# SQL – Egyedi értékek, kulcsok – 2

---

- Bár tudjuk biztosítani, hogy ne lehessen azonos kulccsal bejegyzés, honnan tudjuk, hogy mit kell beszúrni?
  - Megoldás 1 (rossz): megnézzük, hogy mi volt eddig a legnagyobb Id, és eggyel nagyobbát használunk
    - Mi van, ha ezt többen csinálják egyszerre??
  - **Megoldás 2 : Automatikus generálás**
    - MySQL: **AUTO\_INCREMENT**



# SQL – Egyedi értékek, kulcsok – 3

---

## ➤ MySQL: **AUTO\_INCREMENT**

```
CREATE TABLE Termek (
```

```
    Id int PRIMARY KEY AUTO_INCREMENT,
```

```
    Nev nvarchar(50) NOT NULL,
```

```
    Raktarkeszlet int NOT NULL)
```

```
INSERT INTO Termek (Nev, Raktarkeszlet) VALUES ('Alma', 15)
```



# SQL – Összetett kulcsok

➤ Probléma: hogyan garantáljuk a megrendeléstételek egyediségét?

■ Megoldás 1: Külön azonosító (Id) minden rekordhoz

□ MEGRENDELESTETEL (Id, TermekId, MegrendelesId, Db)

□ Probléma: megengedi ugyanazt a terméket többször is, különböző darabszámmal

■ Megoldás 2: összetett kulcs: TermekId+MegrendelesId

```
CREATE TABLE MegrendelesTetel(  
    TermekId int NOT NULL,  
    MegrendelesId int NOT NULL,  
    db int NOT NULL,  
    PRIMARY KEY(TermekId, MegrendelesId)  
)
```



# SQL – Alapértelmezett érték

---

- Ha egy mezőt nem töltünk ki, akkor NULL értéket kap
- Sokszor létezik alapértelmezett érték, használjuk inkább azt
  - Pl új dolgozónál a prémium 0 Ft
  - A számla fizetőeszköze €
- Megoldás: **DEFAULT** kulcsszó

```
CREATE TABLE Termek (  
    Id int PRIMARY KEY AUTO_INCREMENT,  
    Nev nvarchar(50) NOT NULL,  
    Raktarkeszlet int NOT NULL DEFAULT 50)
```

```
INSERT INTO Termek (Nev) VALUES ('Alma')
```



# SQL – Kényszerek

➤ Probléma: a MEGRENDELESTETEL táblába olyan azonosítókat is be lehet szűrni, amelyekhez nem tartozik termék vagy megrendelés.

➤ Megoldás: **Külső kulcs**

- Egy másik reláció kulcsa, csak azok közül kerülhet ki

- **Mező szinten**

- `<mező> <típus> ... REFERENCES <tábla>(<mező2>)`

```
CREATE TABLE MegrendelesTetel(  
    TermekId int NOT NULL REFERENCES Termek(Id),  
    MegrendelesId int NOT NULL REFERENCES Megrendeles(Id),  
    db int NOT NULL,  
    PRIMARY KEY(TermekId, MegrendelesId))
```



# SQL – Kényszerek – 2

## ➤ Megoldás: **Külső kulcs**

### ■ Tábla szinten

□ [CONSTRAINT <név>] FOREIGN KEY (<mező>) REFERENCES <tábla>(<mező<sub>2</sub>>)

### ■ MySQL-ben csak a táblaszintű működik

```
CREATE TABLE MegrendelesTetel(  
    TermekId int NOT NULL,  
    MegrendelesId int NOT NULL,  
    db int NOT NULL,  
    PRIMARY KEY(TermekId, MegrendelesId),  
    CONSTRAINT FK_MT_T FOREIGN KEY (TermekId) REFERENCES Termek(Id),  
    CONSTRAINT FK_MT_M FOREIGN KEY (MegrendelesId) REFERENCES  
        Megrendeles(Id)  
)
```



# SQL – Kényszerek – 3

- Probléma: hogy hivatkozhatunk összetett kulcsra?  
Hogyan tudunk a megrendeléstételekhez pl megjegyzéseket fűzni?

- Megoldás: **Összetett külső kulcs**

- [CONSTRAINT <név>] FOREIGN KEY (<mező1>, <mező2>, ...) REFERENCES <tábla>(<mező1<sub>2</sub>><mező2<sub>2</sub>>, ...)

```
CREATE TABLE MegrendelesTetelMegjegyzes(  
    Id int PRIMARY KEY AUTO_INCREMENT,  
    TermekId int NOT NULL,  
    MegrendelesId int NOT NULL,  
    Megjegyzes nvarchar(1000) NOT NULL,  
    CONSTRAINT FK_MTM_MT FOREIGN KEY (TermekId, MegrendelesID)  
REFERENCES MegrendelesTetel(TermekId, MegrendelesId) )
```



# SQL – Séma módosítás

---

## ➤ Tábla törlése

### ■ DROP TABLE <táblanév>

**DROP TABLE** MegrendelesTetel





# SQL – Séma módosítás – 2

---

## ➤ Új attribútum felvétele

- ALTER TABLE <táblanév> ADD <új oszlop> <típus> <megköt.>

**ALTER TABLE** Termek **ADD** Leiras nvarchar(500) **DEFAULT** '??'

## ➤ Létező attribútum törlése

- ALTER TABLE <táblanév> DROP COLUMN <oszlopnév>

**ALTER TABLE** Termek **DROP COLUMN** Raktarkeszlet



# SQL – Séma módosítás – 3

---

## ➤ Létező attribútum tulajdonságainak változtatása

### ■ MySQL

- ALTER TABLE <táblanév> CHANGE <oszlopnév>  
<új oszlopnév> <új típus> <új megkötések>

**ALTER TABLE** Termek **CHANGE** Nev TeljesNev nvarchar(100) NOT NULL

## ➤ ! Inkompatibilis típusnál hiba vagy adatvesztés !



# SQL – Séma módosítás – 4

---

## ➤ Létező kényszer törlése

- ALTER TABLE <táblanév> DROP FOREIGN KEY <FK\_név>

- Szabvány (MS, ORACLE) : ALTER TABLE <táblanév> DROP **CONSTRAINT** <FK\_név>

```
ALTER TABLE MegrendelesTete1 DROP FOREIGN KEY FK_MT_T
```

## ➤ Új kényszer felvétele

- ALTER TABLE <táblanév> ADD CONSTRAINT <név>  
FOREIGN KEY (...) REFERENCES ...(...)

```
ALTER TABLE MegrendelesTete1 ADD CONSTRAINT FK_MT_T  
FOREIGN KEY (TermekId) REFERENCES Termek(Id)
```



# SQL – Példa – 1

---

- Valósítsuk meg a filmes adatbázist MySQL környezetben!
  - **STUDIÓ** (Id, Név, Székhely)
  - **FILM** (Id, StudióId, Cím, Év, Hossz, Műfaj)
  - **SZÍNÉSZ** (Id, Név, Nem, Születésnap)
  - **SZEREPEL**(FilmId, SzínészId)



# SQL – Példa – 2

---

## ➤ STUDIÓ (Id, Név, Székhely)

```
CREATE TABLE Studio (  
    Id int PRIMARY KEY AUTO_INCREMENT,  
    Nev nvarchar(50) NOT NULL,  
    Szekhely nvarchar(50) NOT NULL)
```



# SQL – Példa – 3

---

## ➤ FILM (Id, StudioId, Cím, Év, Hossz, Műfaj)

```
CREATE TABLE Film (  
    Id int PRIMARY KEY AUTO_INCREMENT,  
    StudioId int NOT NULL,  
    Cim nvarchar(50),  
    Ev int not null,  
    Hossz int not null,  
    Mufaj nvarchar(50),  
    CONSTRAINT FK_Film_Studio FOREIGN KEY (StudioId)  
        REFERENCES Studio(Id)  
)
```

■ Hossz: pl percben kifejezve



# SQL – Példa – 3

---

## ➤ SZÍNÉSZ (Id, Név, Nem, Születésnap)

```
CREATE TABLE Szinesz (  
    Id int PRIMARY KEY AUTO_INCREMENT,  
    Nev nvarchar(100) not null,  
    Nem bit not null,  
    Szuletesnap datetime  
)
```

- Nem: 1 biten tárolható (pl 0 : férfi, 1: nő)
- Születésnap: nem feltétlen ismert



# SQL – Példa – 3

---

## ➤ SZEREPEL(FilmId, SzínészId)

```
CREATE TABLE Szerepel (  
    FilmId int NOT NULL,  
    SzineszId int NOT NULL,  
    PRIMARY KEY(FilmID, SzineszId),  
    CONSTRAINT FK_Szerepel_Film FOREIGN KEY (FilmId)  
        REFERENCES Film(Id),  
    CONSTRAINT FK_Szerepel_Szinesz FOREIGN KEY (SzineszId)  
        REFERENCES Szinesz(Id)  
)
```





# DEMO

---