Progelmélet órák

# 2020. november 17.

DDL utasítások = Data Definition Language

1. **Létrehozás**

**Adatbázis létrehozása:**

**F7**

Álmodból felkelve is tudni kell!!!

**CREATE DATABASE** abneve

**DEFAULT CHARSET =** utf8

**COLLATE** utf8\_hungarian\_ci;

**Tábla létrehozása:**

**CREATE TABLE** tablaneve

Ezt a sort annyiszor ismételjük, ahány sora lesz a táblának.

oszlopneve oszlopadattípusa;

Például:

**CREATE TABLE** Diakok

Diakok

|  |  |  |
| --- | --- | --- |
| diak\_id | diak\_nev | diak\_szul |
| … | … | … |
| … | … | … |

{

diak\_id int,

diak\_nev varchar(30),

diak\_szul date

};

**Mindig** meg kell adni, hogy melyik oszlop legyen az **elsődleges kulcs**. Ezt az utolsó oszlop deklarálása után tehetjük meg. Vagyis a fenti kód

**CONSTRAINT** pk\_Diakok **PRIMARY KEY** (diak\_id)

kulcs neve

kulcsot létrehozó utasítás

megadjuk, hogy milyen kulcs

Annak az oszlopnak a neve, ami a kulcs lesz.

Ha a táblában **idegen kulcs** szerepel, azt is meg kell adni:

Ha a fenti táblában pl. a diak\_nev nevű oszlop az idegen kulcs, és az a Verseny nevű tábla versenyzo\_nev oszlopához kapcsolódik, akkor:

**CONSTRAINT**fk\_Diakok **FOREIGN KEY** (diak\_nev) **REFERENCES** Verseny(versenyzo\_nev)

Megadjuk, hogy melyik tábla melyik oszlopához kapcs.

Elsődleges és idegen kulccsal tehát így kell helyesen létrehozni egy konkrét táblát:

**CREATE TABLE** Diakok

{

diak\_id int,

diak\_nev varchar(30),

diak\_szul date,

**CONSTRAINT** pk\_Diakok **PRIMARY KEY** (diak\_id),

**CONSTRAINT**fk\_Diakok **FOREIGN KEY** (diak\_nev)   
**REFERENCES** Verseny(versenyzo\_nev)

};

# 2020. november 30.

1. **Megszorítások**

Oszlop megszorítások: olyan követelmény, amely behatárolja az adott mezőben (=oszlopban) megjelenő értékek körét.

NOT NULL az adott oszlopban nem szerepelhet üres cella, vagyis egy új sor beszúrása esetén kötelező adatot megadni ehhez a mezőhöz.

UNIQUE az adott oszlopban csak egyedi értékek szerepelhetnek (akkor használjuk, amikor az adott oszlop nem elsődleges kulcs, de biztosítani szeretnénk, hogy ne legyen két egyforma cella az oszlopban).

CHECK utána zj-ben egy logikai kifejezést kell megadni. Segítségével a beszúrás vagy módosítás feltételét adhatjuk meg.

**CREATE TABLE** Diakok

{

diak\_id int NOT NULL, //kötelező diak\_id-t megadni

diak\_nev varchar(30) NOT NULL, // kötelező diak\_nev-et adni

diak\_szul date,

//nem lehet két egyforma taj, ha mégis meg akarnák adni, akkor hibát jelez az aB.

diak\_taj char(9) UNIQUE,

**CONSTRAINT** pk\_Diakok **PRIMARY KEY** (diak\_id),

**CONSTRAINT**fk\_Diakok **FOREIGN KEY** (diak\_nev)

**REFERENCES** tablanev(oszlopnev)

};

DQL utasítások = Data Query Language (adatlekérdező utasítások)

Lekérdezés kulcsszava: SELECT

1. Egy tábla minden oszlopát lekérdezzük (megjelenítjük a képernyőnkön):

**SELECT \* FROM** tablanev;

1. Egy tábla egyetlen oszlopát kérdezzük le:

**SELECT** oszlopnev **FROM** tablanev;

1. Egy tábla 2 vagy több mezőjét kérdezzük le:

**SELECT** oszlopnev1, oszlopnev2, oszlopnev3,….. **FROM** tablanev;

(a megjelenítendő oszlopok neveit vesszővel választjuk el egymástól)

1. A lekérdezett mezőben az egyforma adatok csak egyszer jelennek meg.

**SELECT DISTINCT** oszlopnev **FROM** tablanev;

1. A **WHERE** feltétel

A **WHERE** kulcsszóval feltételt adhatunk meg a lekérdezéshez.

Pl: Jelenítsük meg azoknak a versenyzőknek a nevét és pontszámát, akiknek 700-nál több pontja van:

Ide nem azt kell írni, ami a szövegben van. Hanem a **táblában lévő oszlop** nevét - hajszálpontosan!!!

**SELECT** nev, pont **FROM** versenyzo

**WHERE** pont > 700;

A **WHERE** feltételben: (ez még mindig az 5. pont)

* **Összehasonlító operátorokat** használhatunk ( = , < , > , <= , >= )

Ha az összehasonlítás alapja szöveg, akkor a szöveget aposztrófok közé kell tenni.

Pl: Listázd ki azoknak a tanulóknak a nevét és szakmaazonosítóját, akik Németországból érkeztek (Németország az id-je: GER):

**SELECT** nev, szakmaId **FROM** versenyzo

**WHERE** orszagId = ’GER’;

* **A feltétel lehet összetett**. Ilyenkor az ÉS (AND) valamint a VAGY (OR) művelet használható.

Pl: Jelenítsd meg azoknak a versenyzőknek a nevét és pontszámát, akik 100 pontnál kevesebbet értek el és grafikus a szakmájuk (grafikus szakmaid-je: 40):

**SELECT** nev, pont **FROM** versenyzo

**WHERE** pont <100 **AND** szakmaID = ’40’; (itt a szakmaID nevű oszlop értékei varchar típusúak, ezért aposztrófok közé tesszük.)

**2020. 12.07. (Folytatás)** (ez még mindig az 5. pont)

* **IS NULL / IS NOT NULL**

Feltételként megadva azt vizsgálja, hogy az adott oszlop valamely sora üres-e / nem üres-e.

……… **WHERE** oszlopnev **IS NULL**  ha az oszlopnev nevű oszlopban üres cellát talál

* **LIKE** kulcsszó – azt vizsgálja, hogy a megadott szövegben vannak-e egyező karakterek.
  + % bármennyi és bármilyen karakter lehet a helyén
  + \_ (aláhúzásvonal) egyetlen karaktert keres.

Pl: **SELECT** tnev **FROM** tanulo **WHERE** szhely **LIKE** ’P%’;

Kilistázza a tnev oszlop értékeit (kíírja a tanulók nevét), akiknél a szhely nevű oszlopban: megkeresi azokat a cellákat, amelyekben p betűvel kezdődnek az értékek.

(Poroszló, Parádsasvár, Pécs,……stb)

’%P’  P-re végződik a keresett szó/szöveg

**2020.12.14. (Folytatás)** (ez még mindig az 5. pont)

* **BETWEEN** kulcsszó: keresési intervallum adható meg a segítségével.

Pl: Listázzuk ki a tanulok táblából a tanuló nevét (tnev) és a születési évét (szev), ha 2000 és 2005 között született:

**SELECT** tnev, szev **FROM** tanulok **WHERE** szev **BETWEEN** 2000 **AND** 2005;

* **NOT = tagadás**

Pl: **SELECT** tnev **FROM** tanulok **WHERE NOT** szev = 2015;

**6) Eredménytábla rendezése: ORDER BY**

* A rendezés lehet
  + növekvő: ASC (pl a megjelenő adatokat abc sorrendbe listázza a-tól kezdve)
  + vagy csökkenő: DESC
* Alapértelmezett rendezés: az ASC kulcsszó beírása nélkül is növekvő sorrendbe teszi.

A két sor ugyanazt eredményezi.

Pl: **SELECT** tnev **FROM** tanulok **ORDER BY …**

**SELECT** tnev **FROM** tanulok **ORDER BY …ASC**

* Ha több oszlopot is kilistázunk, akkor több rendezési szempontot is megadhatunk.

Pl: A tanulok nevű táblából listázzunk ki minden sort úgy, hogy születési év szerint csökkenő, azon belül pedig nevek szerint növekvő sorrendbe legyen:

**SELECT** \* **FROM** tanulok **ORDER BY** szev **DESC**, tnev **ASC**;

Az eredménytábla:

Figyeld meg:

A szev oszlop növekvőben van.

A tnev pedig az szev-en belül abc-szerint növekvőben van.

(nem az egész oszlop van abc-sorrendben!!! Hanem a szev oszlopon belül.)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **tnev** | **szev** | **….** | **…** | **…** |
| Bartók Bálint | 2003 | …. | … | … |
| Demény János | 2003 | …. | … | … |
| Meleg Viktor | 2003 | …. | … | … |
| Baráz Laura | 2002 | …. | … | … |
| Orosz Vera | 2002 | …. | … | … |
| Török Bianka | 2002 | …. | … | … |
| Nagy Anna | 2001 | …. | … | … |
| Turó Petra | 2001 | …. | … | … |

1. **Meghatározott számú eredménysor megjelenítése: LIMIT**

A feltételnek megfelelő sorok közül csak annyit listáz ki, amennyit megadunk.

Akkor használjuk, ha van ORDER BY a lekérdezésben.

Pl: Listázd ki az eredmenyek táblából az első három helyezett versenyző nevét (vnev) és pontszámát (pont):

**SELECT** vnev, pont **FROM** eredmenyek **ORDER BY** pont **DESC** **LIMIT** 3;

Magyarázat: első három helyezett: legtöbb pontot elérő 3 tanuló  vagyis a legtöbb pont van legelöl, s utána a kevesebbek…Tehát a pontokat csökkenő sorrendbe kell rendezni (DESC).

**2020.12.15**

1. **ALIAS = mezők elnevezése**

Rövidítése: AS

**SELECT** tnev, szev **FROM** tanulok **WHERE** szev **BETWEEN** 2000 **AND** 2005;

A tnev oszlop neve = tanulónév; szev oszlop neve = születési év:

**SELECT** tnev AS tanulónév, szev AS ’születési év’ **FROM** tanulok

**WHERE** szev **BETWEEN** 2000 **AND** 2005;

Ha az alias egy szóból áll, akkor nem kötelező kitenni az aposztrófot vagy idézőjelet, ha több szóból áll, akkor kötelező!

**2021. 01. 04. SZÁMÍTOTT MEZŐK LÉTREHOZÁSA**

1. A lekérdezésekben használhatunk **matematikai műveleteket**, ugyanúgy, mint az excel-táblázatokban.

Pl: Ha a feladat olyan értéket kér, amelyet két oszlop értékének a különbségével lehet kiszámolni, akkor azt meg lehet oldani. Pl: Életkor kiszámítása:

**SELECT** 2020-szulev **AS** ’Versenyző Életkor’ **FROM** Szemely;

1. **Összesítő függvények** (Aggregáló függvények)

Egyetlen érték lesz az eredmény.

Pl: Egy *Jegyek* nevű mező (oszlop) átlagát szeretnénk megjeleníteni, akkor az összes jegy helyett egyetlen érték jelenik meg.

Fontosabb függvények:

* AVG(oszlopnev) = átlag
* SUM(oszlopnev) = összeg
* COUNT(oszlopnev) = dbszám
* MAX(oszlopnev) = legnagyobb érték
* MIN(oszlopnev) = legkisebb érték

Ha ezeket az összesítő fv-ket használjuk, akkor az eredménybe nem számítanak bele azok a sorok, ahol az érték NULL, vagyis üres.

COUNT(\*) - Ha az összes eredménysor dbszámát szeretnénk visszakapni, akkor ezt kell használni -- vagyis az oszlopnév helyett csillagot írunk a zj-be. Ebben az esetben a NULL értéket tartalmazó sorok is beleszámolódnak.

SZEMELY tábla:

| **az** | **nev** | **telepules** | **cim** | **szuletes** | **belepes** | **fizetes** | **diploma** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Nagy József | Szolnok | Tél u. 23. | 1975-08-25 | 2004-03-01 | 570000 | 1 |
| 2 | Pék Réka | Miskolc | Árny u. 29. | 1977-02-03 | 2006-04-01 | 389000 | 1 |
| 3 | Kis Mária | Debrecen | Vértes u. 20. | 1981-04-01 | 2005-03-27 | 680000 | 1 |
| 4 | Érdek Júlia | Szeged | Bíró u. 25. | 1982-03-08 | 2008-09-01 | 190000 | 0 |
| 5 | Boros Evelin | Szolnok | Barát u. 78. | 1988-02-22 | 2001-01-01 | 850000 | 0 |
| 6 | Lórem Kinga | Szeged | Pilisi út 17. | 1973-08-02 | 2000-01-01 | 878000 | 1 |
| 7 | Fix Béla | Szolnok | Burkus u. 80. | 1991-11-05 | 2008-01-01 | 351000 | 1 |

FONTOS!!!

Példák:

1. Átlagfizetés megjelenítése úgy, hogy a mező fejlécén: az *Átlagfizetés* szó jelenjen meg:

| **Átlagfizetés** |
| --- |
| 63500 |

**SELECT AVG(**fizetes**)** AS Átlagfizetés

**FROM** Szemely;

1. Jelenítsük meg, hogy összesen mennyi bért fizetnek ki. A mező neve Összbér legyen.

| **Összbér** |
| --- |
| 635000 |

**SELECT SUM(**fizetes**)** AS Összbér

**FROM** Szemely;

1. Írjuk ki, hány „Boros” vezetéknevű szerepel a táblában.

| **Boros\_nevűek** |
| --- |
| 1 |

**SELECT COUNT(**nev**)** AS Boros\_nevűek

**FROM** Szemely

**WHERE** nev **LIKE** ’Boros%’;

1. Jelenítsd meg, hogy mennyi volt a legtöbb kifizetett fizetés.

| **MAX(fizetes)** |
| --- |
| 878000 |

**SELECT MAX(**fizetes**)**

**FROM** Szemely;

# 2021. 01.11. Csoportképzés – Group by záradék

**A leggyakrabban összesítő függvények alkalmazásakor használjuk.**

| **az** | **nev** | **telepules** | **cim** | **szuletes** | **belepes** | **fizetes** | **diploma** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Nagy József | Szolnok | Tél u. 23. | 1975-08-25 | 2004-03-01 | 570000 | 1 |
| 2 | Pék Réka | Miskolc | Árny u. 29. | 1977-02-03 | 2006-04-01 | 389000 | 1 |
| 3 | Kis Mária | Debrecen | Vértes u. 20. | 1981-04-01 | 2005-03-27 | 680000 | 1 |
| 4 | Érdek Júlia | Szeged | Bíró u. 25. | 1982-03-08 | 2008-09-01 | 190000 | 0 |
| 5 | Boros Evelin | Szolnok | Barát u. 78. | 1988-02-22 | 2001-01-01 | 850000 | 0 |
| 6 | Lórem Kinga | Szeged | Pilisi út 17. | 1973-08-02 | 2000-01-01 | 878000 | 1 |
| 7 | Fix Béla | Szolnok | Burkus u. 80. | 1991-11-05 | 2008-01-01 | 351000 | 1 |

FONTOS!!!

Példák:

1. Írassuk ki a személyek átlagfizetését települése**nként**! (Ez azt jelenti, hogy minden település legyen felsorolva, és mellette jelenjen meg az az átlagbér – vagyis pl a Szegeden lakó összes személy fizetése átlagolva megjelenik Szeged mellett)

*SELECT telepules, AVG(fizetes) AS Átlagfizu FROM Szemely*

| **telepules** | **Átlagfizu** |
| --- | --- |
| Szolnok | 590333 |
| Miskolc | 389000 |
| Debrecen | 680000 |
| Szeged | 534000 |

*GROUP By telepules;*

1. Jelenítsük meg a diplomák száma **szerinti** átlagfizetéseket.

*SELECT diploma, AVG(fizetes) FROM Szemely*

*Group by diploma;*

| **diploma** | **Átlagfizu** |
| --- | --- |
| 0 | 520000 |
| 1 | 573600 |

**A SELECT után írt oszlopnevek (=mezőlista) mindegyikének szerepelnie kell vagy egy összesítő fv-ben, vagy a group by záradékban!!**

# GROUP BY – HAVING  csoportosítási feltétel

A GROUP BY után használhatunk HAVING záradékot. Erre akkor van szükség, ha a csoportosítás utáni sorok nem mindegyikét kell megjelenítenünk, hanem csak valamely feltételnek eleget tevő sorokat. Ezt a (feladatban mindig megadott) **feltételt** kell a HAVING kulcsszó után beírni.

SELECT nev, COUNT(konyv.id) AS konyvek

FROM konyv INNER JOIN szerzo

ON konyv.szerzoId=szerzo.id

GROUP BY   szerzo.nev HAVING   konyvek > 1

ORDER BY   konyvek DESC,  szerzo.nev ASC;

A HAVING hasonló a WHERE záradékhoz, a különbség az, hogy a WHERE a csoportosítás előtt, minden sort vizsgál a feltétel alapján, a HAVING pedig a csoportképzés után, a csoportok sorain belül végzi a szűrést.

# Táblák összekapcsolása – INNER JOIN

Ha az adatokat két táblából kell lekérdeznünk, akkor az érintett táblákat össze kell kapcsolnunk. Az összekapcsolás kulcsszava: INNER JOIN.

Az összekapcsolandó táblák nevét a kifejezés két oldalára kell megadni:

pl: *versenyzo INNER JOIN szakma* //a versenyzo és a szakma nevű tábla összekapcsolása.

Kötelező megadni, hogy a két tábla mely mezőkön keresztül kapcsolódik össze: ON kulcsszóval:

*versenyzo INNER JOIN szakma ON versenyzo.szakmaId = szakma.id*

Az ON utáni két érték egyike mindig az egyik tábla idegen kulcsa (FK), a másik pedig a másik tábla elsődleges kulcsa, amihez az idegen kulcs kapcsolódik (Mindig figyeld a feladatlap ábráját!!)

**2021. 02. 08.**

**DML (Data Manipulation Language) utasítások**

**SELECT INTO:** a select lekérdezés eredményét elmenti egy új táblába.

Ebben az esetben a lekérdezést a tanultak szerint kell megvalósítani, csak az INTO kulcsszóval meg kell jelölni a létrehozandó új táblának a nevét. FROM elé írjuk!!

pl: SELECT nev, COUNT(konyv.id) AS konyvek INTO ujtablaNeve

FROM konyv INNER JOIN szerzo

ON konyv.szerzoId=szerzo.id

GROUP BY   szerzo.nev;

**INSERT INTO….VALUE:** új sor beillesztése a meglévő táblába.

* Az INSERT INTO után meg kell adni a tábla nevét, amelybe adatot szúrunk be.
* Utána ( )-ben fel kell sorolni az oszlopokat, amelyekbe érték fog kerülni.
* Majd a VALUE kulcsszó után meg kell adni a cellákba kerülő értékeket.

Pl: INSERT INTO szerzo (id, nev, szulEv, nemzetiseg)

VALUE (233, ’Csitári Károly’, 1954, ’magyar’);

| **id** | **nev** | **szulEv** | **halEv** | **nemzetiseg** |
| --- | --- | --- | --- | --- |
| ….. | …… | …… | …… | …… |
| …… | …… | …… | …… | …… |
| 233 | Csitári Károly | 1954 |  | magyar |

A példában egy olyan rekord (sor) lett beszúrva a szerzo nevű táblába, amelyben négy mező (oszlop) adatai vannak megadva. A halEv oszlop értéke ebben a sorban üresen marad:

Ha úgy szúrunk be egy új sort a táblába, hogy minden oszlophoz tartozni fog érték, akkor nem kötelező megadni a mezőneveket:

INSERT INTO szerzo

VALUE (233, ’Csitári Károly’, 1954, 2020, ’magyar’);

| **id** | **nev** | **szulEv** | **halEv** | **nemzetiseg** |
| --- | --- | --- | --- | --- |
| ….. | …… | …… | …… | …… |
| …… | …… | …… |  | …… |
| 233 | Csitári Károly | 1954 | 2020 | magyar |

**UPDATE…SET:** már meglévő rekord (sor) tartalmát módosíthatjuk.

* UPDATE kulcsszó után a tábla nevét adjuk meg, amelyben a sort(sorokat) módosítjuk.
* SET kulcsszó után: oszlopNeve = újÉrték (nézd meg a példát!)
* Majd megadjuk a feltételt, amely az adott sorra vonatkozik.

Pl: UPDATE szerzo SET halEv=2020

WHERE nev = ’Csitári Károly’;

Itt abban a sorban fog módosulni a halEv tulajdonság, ahol a Csitári Károly név szerepel.

Ha egy sorban az értéket törölni szeretnénk (üres cella legyen), akkor:

….SET oszlopNev = NULL…. szerkezetet kell használnunk.

**DELETE:** teljes rekordok (sorok) törlésére alkalmas utasítás.

* A DELETE FROM utasítások után megadjuk a tábla nevét, amelyben módosítani szeretnénk.
* Utána a WHERE feltétellel megadjuk, hogy hol kell módosítani a táblában.

Pl: DELETE FROM szerzo

WHERE nev = ’Mohácsi József’;

* A DELETE FROM tablaNev;

utasítással a tábla megmarad, de minden adat törlődik belőle.

**DDL (Data Definition Language) utasítások**

**CREATE** CREATE TABLE tablaNev… CREATE DATABASE….

**ALTER TABLE….ADD:** új oszlop hozzáadására szolgál egy táblában = az adott tábla szerkezetét módosítja.

* ALTER TABLE kifejezés után táblanevet kell megadni.
* Az ADD kulcsszó után az új oszlop nevét és típusát kell beírni.

Pl: Egy ’megjegyzes’ nevű oszlop beszúrása, amelynek max 50 karakterből állhat az értéke:

ALTER TABLE szerzo

ADD megjegyzes varchar(50);

**ALTER TABLE….MODIFY:**  meglévő oszlop típusának módosítása.

Pl: Ha a szulEv oszlop típusa eddig date típus volt, és módosítani szeretnénk egész szám típusra:

ALTER TABLE szerzo

MODIFY szulEv int;

**ALTER TABLE…..DROP:** oszlop törlése az adott táblából.

Pl: a megjegyzés mező törlése:

ALTER TABLE szerzo

DROP megjegyzes;

**DROP TABLE tablanev:** a megadott tábla törlése az adatbázisból.