Összetett lekérdezések

*Adatbázis-kezelés beadandó*

***Készítette Szórád Annamária & Weller Dániel***

Tartalom  
1. Az adatbázis létrehozása

/A – A táblák létrehozása és az adatok importálása

/B – A táblák tartalma

/C – Példa a táblák tartalmára

/D – Példa az elsődleges és az idegen kulcsok összekapcsolására

/E – A kapcsolati tábla

2. Feladatok

/A – Új adat beszúrása

/B – Összetett lekérdezések

* B/1 – Allekérdezés mint érték szerepel
* B/2 – Többszörösen összetett, ahol az allekérdezés mint érték szerepel
* B/3 – Ahol az allekérdezés egy listával tér vissza (IN)
* B/4 – Ahol az allekérdezés egy listával tér vissza (NOT IN)
* B/5 – ’Minden’ feltétel után kerül az allekérdezés (ALL)
* B/6 – ’Bármely’ feltétel után kerül az allekérdezés (ANY)
* B/7 – Allekérdezés mint tábla
* B/8 – UNION (Full Outer Join) használata

/C- Egyszerű csoportosító lekérdezés (aggrerátum függvénye & INNER JOIN)

* C/1 – Csoportosítás előtti feltétellel
* C/2 – Csoportosítás utáni feltétellel

/D – Választó lekérdezések LEFT JOIN használatával

* D/1 – Csak az egyik tábla adatai
* D/2 – Csoportosítással
* D/3 – Több táblával
* D/4 – Minden adat az egyik táblából

1. Adatbázis létrehozása

1/A

Először létrehoztuk a táblákat, beállítottuk az adatok típusát (int, varchar, date, time, tinyint) majd importáltuk a txt-ket az adatbázisba, összekapcsoltuk az idegen kulcsokat és megjelöltük az elsődleges kulcsokat. A txt-ket <http://informatika.fazekas.hu/erettsegi/emelt-szintu-feladatok/> oldalról szereztük, a 2020 októberi érettségiéből.

1/B

A táblák tartalma:

*diak*

*id ((int, elsődleges kulcs, nincs auto\_increment)*

*nev (varchar(50))*

*osztaly (varchar(20))*

*tevekenyseg*

*id (int, elsődleges kulcs, nincs auto\_increment)*

*nev (varchar(60))*

*iskolai (tinyint)*

*jelentkezes*

*diakid (int, idegen kulcs (diak.id))*

*munkaid (int, idegen kulcs (munka.id))*

*ervenyes (tinyint)*

*elfogadva (tinyint)*

*teljesitve (tinyint)*

*munka*

*id (int, elsődleges kulcs, nincs auto\_increment)*

*tevekenysegid (int, idegen kulcs (tevekenyseg.id))*

*datum (date)*

*kezdes (time)*

*hossz (int)*

*maxletszam(int)*

/D

Példa az idegen kulcsok összekapcsolására:

*ALTER TABLE jelentkezes*

*ADD CONSTRAINT fk\_jelentkezes\_diak FOREIGN KEY (diakid) REFERENCES diak (id);*

Itt a jelentkezés táblában lévő diakid-t kötöttük össze a diak tábla elsődleges kulcsával.

/E

A kapcsolati tábla:

A képen szöveg, képernyőkép, sor, Betűtípus látható

Automatikusan generált leírás

1. Feladatok

2/A

A beillesztést a diák táblában hajtottuk végre, ahol nincs beállítva auto\_increment, ezért meg kellett keresni a maximum id-t és azt növelni, hogy az újonnan beállított diák is egyedi id-val rendelkezzen.

A beillesztés kódja:

*INSERT INTO diak VALUES(*

*(SELECT MAX(d.id)+1*

*FROM diak d),*

*"Szórád András", "3/B");*

2/B

Az összetett lekérdezés amit csináltunk visszaadja a legtöbbet teljesítő diákot.

Az id-t egyenlővé tettük azzal a jelentkezés táblában lévő diakid-val, ahol csoportosítva, és csökkenő sorrendbe helyezve az első helyen áll.

A lekérdezés:

*SELECT d.nev*

*FROM diak d*

*WHERE d.id = (SELECT j.diakid*

*FROM jelentkezes j*

*JOIN munka m ON j.munkaid = m.id*

*WHERE j.teljesitve = -1*

*GROUP BY j.diakid*

*ORDER BY SUM(m.hossz) DESC*

*LIMIT 1*

*);*

2/B/2

A többszörösen összetett lekérdezésünk visszaadja azt az osztályt, ahol az átlagos munkahossz a legnagyobb, és a munkaidejük átlagát. (11/A – 3.6413).

Első lépésben kiválasztottuk a kiírandó adatokat (a legtöbbet teljesítő osztály, az átlag). Ezután Inner Join használatával összekapcsoltuk a táblákat (jelentkezes, diak, munka) és az eredményt osztályok szerint csoportosítjuk.

Másodjára az allekérdezést beleillesztve kiválasszuk azokat az osztályokat, és azoknak az átlagos óraszámát, ahol az megegyezik a legnagyobb átlagos óraszámmal. (Ezt egy újabb allekérdezéssel határozzuk meg). Végül növekvő sorrendbe helyezi és kiírja az első osztályt.

*SELECT t.osztaly, t.avg\_hossz*

*FROM (*

*SELECT d.osztaly, AVG(m.hossz) AS avg\_hossz*

*FROM diak d*

*INNER JOIN jelentkezes j ON j.diakid = d.id*

*INNER JOIN munka m ON m.id = j.munkaid*

*GROUP BY d.osztaly*

*) AS t*

*WHERE t.avg\_hossz = (*

*SELECT MAX(avg\_hossz)*

*FROM (*

*SELECT AVG(m.hossz) AS avg\_hossz*

*FROM diak d*

*INNER JOIN jelentkezes j ON j.diakid = d.id*

*INNER JOIN munka m ON m.id = j.munkaid*

*GROUP BY d.osztaly*

*) AS subquery*

*)*

*ORDER BY t.osztaly;*

2/B/3

Az IN-t használó lekérdezés visszaadja azokat a diákokat akik jelentkeztek a lepkeház felügyeletére. Megvizsgálja, hogy a diák id-ja benne van-e a jelentkezés táblában, majd összekapcsolja a jelentkezés táblát a munka táblával, hogy elérje a tevékenységek táblát. Miután ez megvan, csak azokat rakja bele a listába, ahol a tevékenységek között szerepel a „Lepkeház felügyelete”

*SELECT distinct d.nev*

*FROM diak d, tevekenyseg t*

*WHERE d.id IN (*

*SELECT j.diakid*

*FROM jelentkezes j*

*JOIN munka m ON j.munkaid = m.id*

*JOIN tevekenyseg t ON m.tevekenysegid = t.id*

*WHERE t.nev = 'Lepkeház felügyelete' )*

2/B/4

A NOT IN-t használó lekérdezésünk kiválasztja az összes olyan diákot, akiknek nincs teljesített közösségi szolgálata.

*SELECT d.nev*

*FROM diak d*

*WHERE d.id not IN (*

*SELECT j.diakid*

*FROM jelentkezes j*

*WHERE j.teljesitve = -1*

*)*

2/B/5

Az ANY-t arra használtuk, hogy kiválasszuk azokat a diákokat akik több órát dolgoztak le mint bármelyik 11/C-s diák. Ehhez összekötöttük az idegen kulcsokat az elsődleges kulcsokkal a diák, munka, és a jelentkezés táblákban. majd relációt bevezetve összehasonlítottuk a munka tábla hossz oszlopát bármely 11/C-s diákkal, akiket egy allekérdezésben választottunk ki.

*SELECT distinct d.nev*

*FROM diak d, munka m, jelentkezes j*

*WHERE j.diakid = d.id AND j.munkaid = m.id AND m.hossz > ANY (SELECT m.hossz FROM munka m, diak d WHERE d.osztaly = '11/C');*

2/B/6

A következő feladathoz az ALL parancsot használtuk annak érdekében, hogy kiválasszuk az összes olyan diákot, akik 2017. decembere előtt kerültek be a jelentkezés táblába. Ehhez össze kellett kapcsolni a táblákat, majd egy allekérdezésben kiválogatni a dátum alapján az adatokat, végül a reláció megfelelő végére raktuk.

*SELECT DISTINCT d.nev*

*FROM diak d, munka m, jelentkezes j*

*WHERE j.diakid = d.id AND j.munkaid = m.id AND m.datum > ALL (SELECT m.datum FROM munka m WHERE m.datum = '2017-12-01');*

2/B/7

A táblaként használt allekérdezésünk azt adja meg, hogy egy bizonyos dátumon hány fajta tevékenység folyt. (a példában 2017.08.31 szerepel, ahol a betegirányítás és a gyermekfelügyelet volt).

A lekérdezés a "munka" táblához csatlakoztatott egy al-lekérdezést, amely kiválasztotta az elfogadott és teljesítetlen jelentkezések munkaazonosítóit. Ezután a "munka" táblához csatlakoztattuk az al-lekérdezést és kiválasztottuk a megfelelő dátumot. A "diak", "jelentkezes" és "tevekenyseg" táblákat összekapcsoltuk a megfelelő kapcsolatokon keresztül.

Végül a kiválasztott sorok közül csak azokat tartottuk meg, ahol a dátum '2017-08-31' volt. Az eredmény a tevékenységek neveivel és a hozzájuk rendelt dátummal tér vissza, de csak az egyedi értékeket tartalmazza.

*SELECT DISTINCT m.datum, t.nev*

*FROM munka m*

*INNER JOIN(*

*SELECT j.munkaid*

*FROM jelentkezes j*

*WHERE j.elfogadva = -1 AND j.teljesitve = -1*

*) AS j ON m.id = j.munkaid, diak d, jelentkezes j, tevekenyseg t*

*WHERE m.tevekenysegid = t.id and j.diakid = d.id and m.datum = '2017-08-31';*

2/B/8

A lekérdezés visszatért a diákok nevével, osztályával és a tevékenységek nevével, ahol teljesültek a következő feltételek:

Csatlakoztattuk a "diak" táblát a "jelentkezes" táblához balra a diák azonosítója alapján, majd a "jelentkezes" táblát a "munka" táblához a munka azonosítója alapján. Végül összekapcsoltuk a "munka" táblát a "tevekenyseg" táblával a tevékenység azonosítója alapján.

A lekérdezés második részében a "tevekenyseg" táblát csatlakoztattuk balra a "jelentkezes" táblával a tevékenység azonosítója alapján. Azonban itt nem tartottunk igényt diák információra, ezért a "NULL" értékeket adtuk vissza a "nev" és "osztaly" oszlopokban. Csak azokat a tevékenységeket tartottuk meg, ahol az "iskolai" érték 0, és a "jelentkezes" táblában a "teljesitve" értéke -1 vagy NULL.

Ezáltal a lekérdezés visszaadta a diákok nevét, osztályát és a tevékenységek nevét azokra a diákokra és tevékenységekre vonatkozóan, amelyek megfeleltek a feltételeknek.

*SELECT d.nev AS nev, d.osztaly AS osztaly, t.nev AS tevekenyseg\_nev*

*FROM diak d*

*LEFT JOIN jelentkezes j ON d.id = j.diakid*

*LEFT JOIN munka m ON j.munkaid = m.id*

*LEFT JOIN tevekenyseg t ON m.tevekenysegid = t.id*

*WHERE t.iskolai = -1*

*UNION*

*SELECT NULL AS nev, NULL AS osztaly, t.nev AS tevekenyseg\_nev*

*FROM tevekenyseg t*

*LEFT JOIN jelentkezes j ON t.id = j.munkaid*

*WHERE t.iskolai = 0*

*AND (j.teljesitve = -1 OR j.teljesitve IS NULL);*

2/C

A csoportosító lekérdezésünk visszaadja osztályokra bontva azt, hogy egy osztályban hány órát teljesítettek már.

Csatlakoztattuk a "diak" táblát a "jelentkezes" táblához belső csatlakozással a diák azonosítója alapján, majd a "jelentkezes" táblát a "munka" táblához belső csatlakozással a munka azonosítója alapján.

Csak azokat az eredményeket tartottuk meg, ahol a jelentkezés "teljesitve" értéke -1 volt, ami azt jelenti, hogy a munkát elvégezték.

Az eredményeket a diákok osztály szerint csoportosítottuk.

Végül az összesített munkahosszt számoltuk az egyes osztályokhoz tartozó diákok számára.

*SELECT d.osztaly, SUM(m.hossz) AS osszeg*

*FROM diak d*

*INNER JOIN jelentkezes j ON d.id = j.diakid*

*INNER JOIN munka m ON j.munkaid = m.id*

*WHERE j.teljesitve = -1*

*GROUP BY d.osztaly;*

2/C/2

A második csoportosító lekérdezés visszatért a diákok osztályával és a munkák hosszának összegével.

Csatlakoztattuk a "diak" táblát a "jelentkezes" táblához belső csatlakozással a diák azonosítója alapján, majd a "jelentkezes" táblát a "munka" táblához belső csatlakozással a munka azonosítója alapján majd az eredményeket a diákok osztálya szerint csoportosítottuk.

A csoportosított eredményeket a "HAVING" feltétellel szűrtük,ott, ahol az összesített munkahossz kevesebb volt, mint 450.

Így a lekérdezés visszatért a diákok osztályával és a munkák hosszának összegével azokra a diákokra vonatkozóan, akiknek az összesített munkahossza kevesebb volt, mint 450.

*SELECT d.osztaly, SUM(m.hossz) AS osszeg*

*FROM diak d*

*INNER JOIN jelentkezes j ON d.id = j.diakid*

*INNER JOIN munka m ON j.munkaid = m.id*

*GROUP BY d.osztaly*

*HAVING SUM(m.hossz) < 450;*

2/D

*SELECT d.nev*

*FROM diak d*

*LEFT JOIN jelentkezes j ON d.id = j.diakid*

*WHERE j.diakid IS NULL;*

2/D/2

*SELECT d.osztaly, COUNT(j.diakid) AS diak\_szam*

*FROM diak d*

*LEFT JOIN jelentkezes j ON d.id = j.diakid*

*GROUP BY d.osztaly;*

2/D/3

*SELECT d.nev, t.nev AS tevekenyseg\_nev*

*FROM diak d*

*LEFT JOIN jelentkezes j ON d.id = j.diakid*

*LEFT JOIN munka m ON j.munkaid = m.id*

*LEFT JOIN tevekenyseg t ON m.tevekenysegid = t.id;*

2/D/4

*SELECT t.nev AS tevekenyseg\_neve, COUNT(d.id) AS diakok\_szama*

*FROM tevekenyseg t*

*LEFT JOIN munka m ON t.id = m.tevekenysegid*

*LEFT JOIN jelentkezes j ON m.id = j.munkaid*

*LEFT JOIN diak d ON j.diakid = d.id*

*GROUP BY t.nev;*