Adatbázis-kezelés

Koloszár László
Tóth Zsolt

Adatbázis-kezelés

írta Koloszár László és Tóth Zsolt

KÉSZÜLT A TÁMOP-4.1.2.A/1-11/1-2011-0096 TÉMÁJÚ, FELSŐFOKÚ SZAKKÉPZÉSEK FEJLESZTÉSE AZ NYME TTK-N PROJEKT KERETÉBEN

Tartalom

1. fejezet - Bevezetés

Adatbázis-kezelés jegyzetünk elsősorban a felsőfokú szakképzés keretében folyó integrált gazdasági és informatikai jellegű képzések hallgatói számára készült.

A jegyzet írása során arra törekedtünk, hogy a tananyagot a célcsoport igényeihez szabjuk. Úgy véljük, hogy a hallgatók későbbi adatbázis-fejlesztési munkájuk során nem nélkülözhetik az elméleti alapokat, de a jegyzetben a műszaki felsőoktatás szintjéhez képest rövidebben tárgyaljuk ezeket, mivel a célcsoport főként asszisztensi és nem tervezői szerepkörben fog tevékenykedni. Az adatbázis-kezelés elméletét csak olyan mértékben részletezzük, amennyire informatikával foglalkozó, nem kimondottan műszaki szakembereknek szükségük lehet.

A rendszerszintű, holisztikus szemlélet e feladatkörökben is kiemelt erény, a jegyzet e nézőpont érvényesítésére törekszik, ugyanakkor a gyakorlatorientált feldolgozás kiemelt hangsúlyt kap. Ez nem csak a megcélzott hallgatókhoz hozza közelebb a témakört, hanem reményeink szerint az önálló, kreatív, ötletgazdag és proaktív munkavégzést is segíteni fogja.

A szoftverhátteret az adatbázis-kezelési alapozáshoz választottuk. Olyan szoftvert kerestünk, amely a kisebb adatbázisok számára optimális, jól paraméterezhető és rendkívül felhasználóbarát. Ezáltal lehetőség van az adatbázis-kezelés logikai összefüggéseinek a konkrét szoftvertől függetlenelsajátítására is, miközben a kezdő hallgatók nem rettennek meg a keretrendszertől. A jegyzet így a mélyebb ismeretek megszerzéséhez kiváló alapot ad, a további feldolgozás akár már önálló hallgatói feladatvégzéssel is megtörténhet. Választásunk a fenti elvárásokhoz talán legközelebb álló Microsoft Access szoftverre esett.

A jegyzet elméleti bevezetőjét és az Access használatának alapjait tartalmazó első részét Tóth Zsolt, a gyakorlati példákat és a megoldásokat tartalmazó második részét Koloszár László készítette.

Reméljük, hogy a jegyzet hasznos segítséget jelent adatbázis-kezelési tanulmányaik során.

2. fejezet - Elméleti háttér

Az adatbázis fogalma és az adatbázisok rövid története

Az *adatbázis* fogalmának számos egzakt, tudományos definíciója létezik. Számunkra azonban tökéletesen elégségesnek tűnik az a hétköznapi definíció, amely szerint az adatbázis valamilyen jól definiált rendszer szerint tárolt adatokból áll, s lehetővé teszi az adatok kezelését, azaz rögzítését, tárolását, rendszerezését, keresését, módosítását, különböző kimutatások és lekérdezések készítését stb. Lényegében az ezeket a funkciókat biztosító szervezettség teszi az adathalmazt adatbázissá. Tágabb értelemben az adatbázisok funkcióit biztosító rendszert nevezzük adatbázis-kezelőnek, az ezzel foglalkozó diszciplínát pedig adatbázis-kezelésnek.

Az adatok kezelését lehetővé rendszerek természetesen a számítástechnika kialakulása előtt is léteztek. Az ősi folyamvölgyi kultúrák, a görög poliszok, a római birodalom, a középkori egyházi, nemesi és királyi levéltárak, könyvtárak, adóügyi és birtok-nyilvántartások, az újkori államok és egyéb szervezetek széleskörű nyilvántartási rendszereinek stb. története nemcsak tudománytörténeti szempontból fontos. Az akkori problémákra adott válaszok esetenként mai körülményeink között is érdekesek és hasznosak.

Mi azonban e jegyzet keretében csak a számítástechnikai eszközökkel végzett adatbázis-kezeléssel tudunk érdemben foglalkozni.

Az adatok felvételének, tárolásának, rendszerezésének és visszakeresésének gépesítése a 19. század második felében vált egyre sürgetőbb igénnyé.

Az első adatbázis-kezelő, mechanikus gépnek *Herman Hollerith* német származású amerikai feltaláló lyukkártyás rendszere tekinthető, amelynek a használata az 1890-es amerikai népszámlálás adatfelvételét és adatfeldolgozását jelentős mértékben meggyorsította.

Az adatbázis-kezelés valódi forradalmát azonban a számítástechnikai eszközökkel végzett adatbázis-kezelés kialakulása és gyors fejlődése jelentette. A hőskor lyukkártyaolvasó és lyukkártyaíró rendszereit igen hamar felváltották a mágnesszalagos, majd optikai tárolók, s általában véve minden számítástechnikai-infokommunikációs eszköz fejlődésének igen komoly hatása volt az adatbázis-kezelő rendszerekre. A hálózatok, majd az internet fejlődése pedig az osztott rendszerektől az adatbázis-kapcsolatos internet-alkalmazásokig az adatbázis-kezelés újabb ugrásszerű fejlődését hozta magával. Ma a különböző online alkalmazások felhasználóinak többsége valószínűleg nincs is tisztában azzal, hogy a napi, intenzív webes kommunikáció során az online alkalmazás a háttérben megbújó, kifinomult adatbáziskezelő-alkalmazásokat használ.

Talán paradoxonnak tűnhet, de a mai adatbázisok és adatbázis-kezelő rendszerek elvi alapjai lényegében évtizedek óta alig-alig változtak. Az infokommunikációs eszközök és környezet változása miatt ugyan a mai diákok számára egy évtizedekkel ezelőtti számítógép és számítógépes alkalmazás használhatatlan „őskövületnek” tűnhet, az adatbázis-kezelés standard, s az adatbázisok fejlesztéséhez nélkülözhetetlen elmélete nem avult el az évtizedek során.

Adatmodellek

Az adatbázis-kezelés elméletének magja az *adatmodell*, az adatbázisban tárolt adatok adatszervezési módjának sémája. Több elterjedt modell létezik, de az adatbázisok többsége a relációs adatmodellre épül. A jegyzet első felét alapvetően e modellnek a bemutatására szenteljük.

Mielőtt azonban részletesen megismerkednénk a relációs modellel, röviden bemutatunk néhány egyéb adatmodellt is.

Hierarchikus adatbázismodell

A hierarchikus adatbázisokat általában faszerkezettel szokás szemléltetni. Hierarchikus adatmodellt követ például a számítógépek fájlszerkezete vagy a Windows regisztrációs adatbázisa.

A hierarchikus adatbázisokban az adatok hierarchikus rendbe szervezik. Az alárendelt adatokat *gyerekadatnak*, a fölérendelteket *szülőadatnak* is nevezik. A fölé- és alárendelt adatok között általában „egy a többhöz” kapcsolat van, azaz a szülőadatnak több gyerekadata lehet, de fordítva ez már nem teljesül, a gyerekadatok csak egy szülőadattal rendelkeznek.

Az adatok elérése során relatív elérésről vagy relatív elérési útról beszélünk, amikor a hierarchia egy adott helyén lévő adatból kiindulva írjuk le egy másik adat helyét a kapcsolatokon keresztül. (Például: ../../első szint/második szint/adat)

Az adatok elérése abszolút, amikor egy adat elérési helyét a hierarchia legfelsőbb szintjéről kiindulva írjuk le. (Például: C:/első szint/második szint/adat)

Hálós adatbázismodell

A hálós adatmodellben lényegében a matematikából ismert gráfokkal írjuk le az adatbázist. Újabban elsősorban a digitális térképfeldolgozásnál értékelődött fel – az előző modellhez hasonlóan – nem túl széles körben használt adatmodell.

A hálós adatmodellben az adatok az adatháló csomópontjaiban helyezkednek el. Az adatok közötti kapcsolat (a háló éleinek) minőségét az adatok közötti kapcsolat hossza is jellemezheti. Az adatok között a kapcsolat irányított vagy irányítatlan lehet.

Többdimenziós adatmodell

A döntés-előkészítésben használt *adatbányász* rendszerek gyakran többdimenziós adatmodellre épülnek. Az adatokat általában egy többdimenziós „kockában” tárolják, s ez a tárolási mód lehetőséget ad gyakorlatilag bármilyen típusú adatkombináció és adatösszesítés lekérdezésére. A többdimenziós adatmodell elsősorban a nagy adatbázisok statisztikai elemzését teszi lehetővé.

Objektumrelációs adatmodell

Az objektumrelációs adatmodell alapvetően a relációs modell összetett adattípusokkal bővített változatának tekinthető.

Az adatbázis-kezelés standard alapmodellje azonban a relációs adatmodell, amelyet szinte egyeduralkodó jellege miatt részletesebben is bemutatunk.

A relációs adatmodell

A relációs modell szülőatyja *Edgar Frank Ted Codd*, aki 1970-ben publikált *A Relational Model of Data for Large Shared Data Banks* (A nagy, osztott adatbankok egy lehetséges relációs adatmodellje) című cikkében halmazelméleti alapokon, igen tömören és pontosan (mindössze 11 oldalon) vázolta fel azt az adatmodellt, amely több mint negyven éve szinte minden adatbázis-kapcsolatos alkalmazás megkerülhetetlen része.

A modell alapját képező *relációs algebra* a relációs adatbázisok elvi és gyakorlati problémáinak megoldására alkalmas módszertan, mi azonban ezzel a jegyzet keretei között nem foglalkozunk.

Az *IBM* (többek között Codd közreműködésével) 1976-ban kifejlesztette az első, relációs algebrára épülő *adatbázisnyelvet*, a *SEQUEL*-t, amely később a rövidebb *SQL*-re (*Structured Query Language*, strukturált lekérdezőnyelv) neveztek át.

Az SQL segítségével lehetővé vált, hogy a fejlesztők szabványos adatszerkezeteket definiáljanak, az adatkezelés bonyolultsága csökkent, s az adatbiztonsági problémák megoldása is könnyebb lett. Az SQL-ben a lekérdezések és a karbantartás (adatfelvétel, adatmódosítás, adattörlés) során már nem kellett a fizikai szinttel foglalkozni.

A relációs modell és a folyamatosan fejlődő SQL-nyelv – az informatikai fejlődés több más eleme mellett – tette lehetővé a mai adatbázis-kezelő rendszerek kifejlesztését, amely többségükben az a relációs adatmodellre épülnek, és a kezelőfelületen végzett műveletek „alatti” szinten SQL-utasítások futnak.

Operatív és analitikus adatbázisok

Az adatbázisokat alapvetően két csoportra oszthatjuk: *operatív* és *analitikus* adatbázisokra.

Az operatív adatbázisok az üzleti életben, a különböző intézményekben és szervezetekben széles körben elterjedtek. Az operációs adatbázisokban a különböző módon összegyűjtött adatok tárolását, kezelését és módosítását végezzük. Az adatok változása miatt azokat dinamikus adatoknak tekintjük.

Az analitikus adatbázisok viszont statikus adatokat tárolnak. Ezek lehetnek például tranzakciók adatai vagy történeti statisztikai adatok.

Bár a kétféle adatbázis eltérő követelményeket támaszt a fejlesztő és a felhasználó felé, s ma már az elvi modellekben is egyre jelentősebbek a különbségek, Codd relációs modellje alapvetően mindkét adatbázistípusnál alkalmazható.

Alapfogalmak

A relációs modellben az adatokat *relációkban* vagy *táblákban* tároljuk. A két fogalom minden további nélkül felcserélhető egymással. Fontosnak tartjuk megjegyezni, hogy a relációt a hétköznapi szóhasználatban „kapcsolatként” fordítjuk, ezért a relációt nagyon sokan a táblák közötti kapcsolattal azonosítják. Ez azonban hibás értelmezés. A reláció nem a táblák közötti kapcsolatot, hanem magát a táblát jelenti.

A táblát (relációt) *egyednek* is nevezhetjük. Az egyed lényegében a tábla tartalmi megfelelője, hiszen az egyed olyan *egyedpéldányok* halmaza, akik a többi egyedtől és egyedpéldánytól (dologtól, objektumtól) jól elkülöníthetők, megkülönböztethetők.

Minden tábla sorokból és oszlopokból áll.

A sorokat gyakran hívják *rekordnak*, tartalmilag pedig egy-egy egyedpéldánynak felelnek meg. A sorok sorrendje tetszőleges, és a táblában nem lehet két olyan sor, amelyiknek mindegyik eleme megegyezne.

Az oszlopok alternatív elnevezése a *mező*. Az oszlop tartalmilag *tulajdonságként* írható le.

A sorok és az oszlopok metszetei *tulajdonságértékeket, elemi adatokat* tartalmaznak.

A *kulcs* olyan oszlop, amely speciális célokkal-tulajdonságokkal rendelkezik.

Az *elsődleges kulcs* egy olyan kitüntetett tulajdonságokkal rendelkező oszlop, amely egyértelműen azonosítja, s egyben megkülönbözteti egymástól a sorokat. Ennek valójában egyetlen elégséges és szükséges feltétele van: az elsődleges kulcsként funkcionáló oszlopban minden egyes értéknek különböznie kell. Bár a használatuk különböző okokból nem célszerű, léteznek ún. *összetett kulcsok* is, amikor az elsődleges kulcsot nem egy, hanem több oszlop alkotja. Ilyenkor a kettő vagy több oszlop akkor lehet elsődleges kulcs, ha nincs két olyan sor, amelyekben az elsődleges kulcsokhoz tartozó oszlopok összes értéke megegyezne.

Az egyik táblában összetett kulcsként definiált oszlopok egyes oszlopaihoz egy másik táblában tartozhat egy *idegen kulcsként* funkcionáló oszlop. Az idegen kulcs elemei az első tábla elsődleges kulcsának

elemei közül kerülnek ki, s ezzel az idegen kulcs a két tábla közötti kapcsolatot testesíti meg. Az idegen kulcsként definiált oszlop azonban a táblán belül általában egy egyszerű, nem elsődleges kulcsként funkcionáló oszlop.

A kapcsolatok típusai

A több táblába rendezett adatok közötti kapcsolatoknak három típusát lehet megkülönböztetni: az *egy-az-egyhez* kapcsolatot (1:1), az *egy-a-többhöz* kapcsolatot (1:n) és a *több-a-többhöz* kapcsolatot (n:m).

Az egyes kapcsolattípusok leírásához először képzeljük el, hogy az adathalmazunk egyedei közül kiválasztunk két egyedet, az első egyed egyedpéldányai alkotják az *A halmazt*, a második egyed egyedpéldányai a *B halmazt*. (Az egyedekre az egyes kapcsolattípusoknál konkrét példákat adunk.) A halmazok természetesen táblaként is értelmezhetők.

Egy-az-egyhez kapcsolat

Egy-az-egyhez kapcsolatnál A halmaz (tábla) minden eleméhez (sorához) B halmaz (tábla) legfeljebb egy eleme (sora) tartozik, és ugyanígy, B halmaz (tábla) minden eleméhez (sorához) A halmaz (tábla) legfeljebb egy eleme (sora) tartozik.

Az egy-az-egyhez kapcsolatban lévő táblák valójában egyetlen táblaként is kezelhetők lennének. Az adatokat egy-az-egyhez kapcsolat esetén általában azért bontjuk két táblára, mert például a túlságosan összetett táblát több kisebb, könnyebben kezelhető táblára kívánunk felosztani, vagy egy tábla egyes részeit adatvédelmi okokból külön kívánjuk tárolni, vagy az egyik táblában olyan adatokat kívánunk tárolni, amely a főtáblában tartalmilag csak egyes soroknál értelmezhető.

Példa: Létrehozhatunk egy táblát azokról a hallgatókról, akik szabadidejükben az egyetemi kosárlabdacsapat oszlopos tagjai.

Egy-a-többhöz kapcsolat

Az egy-a-többhöz kapcsolat a leggyakoribb kapcsolattípus. Egy-a-többhöz kapcsolatnál A halmaz (tábla) minden eleméhez (sorához) B halmaz (tábla) legfeljebb egy eleme (sora) tartozik, de a korlát fordítva már nem igaz, B halmaz (tábla) minden eleméhez (sorához) A halmaz (tábla) több eleme (sora) tartozhat.

Példa: Vallásos muszlim férfinak nem szekularizált országban több felesége lehet, de a nők csak egy férfihez mehetnek hozzá feleségül.

Több-a-többhöz kapcsolat

Több-a-többhöz kapcsolatnál A halmaz (tábla) minden eleméhez (sorához) B halmaz (tábla) több eleme (sora) tartozhat, és ugyanígy, B halmaz (tábla) minden eleméhez (sorához) A halmaz (tábla) több eleme (sora) rendelhető hozzá. A több-a-többhöz kapcsolat technikailag minden esetben felbontható két egy-a-többhöz kapcsolatra.

Példa: A hallgatókat több oktató tanítja, s az oktatók is több diáknak tartanak órát.

Adatbázisok tervezése

Egy megfelelően működő adatbázis elképzelhetetlen a nagyon pontosan kivitelezett tervezési szakasz nélkül. Különösen nagyobb adatbázisoknál a későbbi, radikális átalakítási igény számos problémát felvethet, ezért már a kezdetektől hatékony tervezésre kell törekedni. Ehhez tisztázni kell néhány alapvető fogalmat és az adatbázis-tervezés célszerű menetét.

Adatmodell és adatbázis

Az adatbázis lényegében az adatmodell konkrét megvalósításának tekinthető. Az adatbázisoknak azonban több absztrakciós szintjük is létezik, mi most ezek közül hármat emelünk ki:

- *Fizikai szint:* Az adatbázisok fizikai szintjén az adatok adattárolókon való elhelyezkedését értjük. Ezzel az absztrakciós szinttel a jegyzetben nem foglalkozunk.
- *Fogalmi-logikai szint:* Az adatbázis egészének leírása az adatmodellre építve. Ezzel a szinttel már – a relációs modellre építve – részletesebben foglalkozunk. Megjegyezzük, hogy a két szintet több szerző elkülöníti egymástól, de mi ezt gyakorlati szempontból nem látjuk szükségesnek.
- *Felhasználói szint:* Az adatbázis-kezelő rendszerek által a felhasználó konkrét igényeinek megfelelő nézet, kezelőfelület. Az adatbázis-kezelés keretében értelemszerűen ezzel is részletesen foglalkozni kell.

Adatbázisséma és adatbázis-előfordulás

A tervezés során, amikor az adatbázis fogalmi szintjének kialakítását előkészítjük, meg kell tervezni az adatbázis szerkezetét, az egyedeket, tulajdonságokat, elsődleges és idegen kulcsokat és kapcsolatokat tartalmazó *adatbázissémát*. Az adatbázisséma a későbbiekben ugyan bővíthető és módosítható, de törekedni kell arra, hogy az esetleg idő- és munkaigényes átalakítás ne tervezési hibák, hanem az újabb igények miatt következzék be.

Az *adatbázis-előfordulás* az adatbázis aktuális tartalmát jelenti. Az alapadatok első feltöltése általában az adatbázisséma alapján lezajló tervezés után történik, frissítésük pedig folyamatos. Minél nagyobb az adatbázis-előfordulás annál nagyobb figyelmet kell szentelni az adatbiztonsági kérdéseknek.

Az adatbázisséma jelölése

Az adatbázisséma jelölésére számos modell létezik. Némileg hagyománytiszteletből szinte minden jegyzet részletesen bemutatja az egyed-kapcsolat modellt, mi azonban ezt nem látjuk szükségesnek.

A sokféle jelölési módszerből a következő, nagyon egyszerű – a relációs adatbázismodellnek leginkább megfelelő – jelölési módszert választjuk:

- A táblák nevét nagybetűvel írjuk.
- A táblanév mögött az egyes tulajdonságokat (oszlopokat) a későbbi mezőneveknek megfelelően zárójelben felsoroljuk.
- Az elsődleges kulcsot félkövérrel szedjük.
- Az idegen kulcsokat csillaggal jelöljük.

A fentiek alapján tehát egy tábla általános sémája:

```
TÁBLANÉV (Elsődleges kulcs, Azonosító 1., Azonosító 2.*,
                    Azonosító 3. … Azonosító n.)
```

A fenti modellnek természetesen hátránya, hogy például a kapcsolattípusokat nem jelöli vizuális eszközökkel, viszont ha tisztázzuk, hogy a relációs adatmodellben hogyan biztosítjuk a különböző típusú kapcsolatokat a táblák között, akkor néhány táblás modelleknél ebből aligha származhatnak problémák. A tábla fenti jelölése ráadásul már a kezdetektől absztrakcióra kényszeríti a kezdő tervezőt, s ez a későbbiekben hasznos lehet.

Kapcsolatok biztosítása a táblák között

Egy-az-egyhez kapcsolat

Az egy-az-egyhez kapcsolatot két tábla között úgy biztosítjuk, hogy a két táblában van egy közös mező.

Például ha az egyetemi hallgatók egy részét külön táblába szerepeltetjük, mert játszanak az egyetemi kosárlabdacsapatban, akkor a két – leegyszerűsített – tábla felépítése a következő:

HALLGATÓK (**Hallgatókód***, Vezetéknév, Karkód, Szakkód …)

KOSÁRLABDÁZÓK (**Hallgatókód***, Magasság, Pozíció …)

A *Hallgatókód* mindkét táblában elsődleges kulcs, de egyben idegen kulcs is..

A *Karkódot* és a *Szakkódot* meg is csillagozhattuk volna, hiszen valószínűleg – a *Kar* és a *Szak* táblázatot feltételezve – idegen kulcsként funkcionálnak.

A közös mezőre vonatkozó feltétel nem a mezőnév azonosságát jelenti, hanem az adatok azonosságára – pontosabban inkább halmaz-részhalmaz viszonyra – utal. Ha a Kosárlabdázók táblában a Hallgatókód helyett például *Kosárlabdázókódot* írtunk volna azonos adatok mellett, a kapcsolat akkor is egy-az-egyhez lett volna. az adatok azonosságára

Egy-a-többhöz kapcsolat

Az egy-a-többhöz kapcsolatot úgy jelöljük, hogy az „egy” oldalon álló tábla (amely tábla értékeihez a másik táblában több is tartozhat) elsődleges kulcsát idegen kulcsként szerepeltetjük a másik, „több” oldalon álló táblában.

Például ha egy nem szekularizált, muszlim ország hadseregének nyilvántartásában két külön táblában szerepeltetik a katonákat (akik az egyszerűség kedvéért, nem egészen a valóságnak megfelelően, legyenek mind férfiak), egy másik táblában pedig a feleségeiket, akkor a két tábla leegyszerűsített sémája a következő:

KATONÁK (**Katonakód**, Vezetéknév, Keresztnév, Egység, Rang …)

FELESÉGEK (**Feleségkód**, Vezetéknév, Keresztnév, Katonakód* …)

Látható, hogy a két tábla közötti egy-a-többhöz kapcsolatot úgy biztosítottuk, hogy a Katonakód elsődleges kulcsot a Feleségek táblában idegen kulcsként tüntettük fel.

Több-a-többhöz kapcsolat

Két tábla között a több-a-többhöz kapcsolatot ún. *illesztőtábla* segítségével biztosítjuk. Az illesztőtáblában mindkét tábla elsődleges kulcsát felvesszük úgy, hogy a két idegen kulcs együtt az illesztőtábla elsődleges (összetett) kulcsát alkotja.

A következő két (plusz egy) táblából álló séma az oktatók és a hallgatók közötti tantárgyi kapcsolatot írja le:

OKTATÓK (**Oktatókód**, Vezetéknév, Keresztnév, Karkód …)

HALLGATÓK (**Hallgatókód**, Vezetéknév, Keresztnév, Karkód, Szakkód …)

TANTÁRGYI KAPCSOLAT (**Oktatókód*, Hallgatókód*** …)

Látható, hogy a két tábla közötti több-a-többhöz kapcsolatot úgy biztosítottuk, hogy az Oktatókód és a Hallgatókód a Tantárgyi kapcsolat illesztőtáblában egyenként idegen kulcsként, együtt összetett kulcsként szerepel.

Anomáliák

Az adatbázisok használata – adatokkal való feltöltése – közben felléphetnek ún. *anomáliák*, problémák, amelyek megnehezítik a táblák használatát.

A legfontosabb anomáliák a következők:

- *Redundancia*: Az információk több sorban feleslegesen ismétlődnek.
- *Beszúrási probléma*: Az adathoz kötelezően kapcsolódik egy másik adat, de azt nem ismerjük, ezért az adatot nem tudjuk bevinni a táblába.
- *Módosítási probléma*: Módosítani kell egy adatot egy sorban, de az adat több más sorban is szerepel, s mégsem módosulnak külön beavatkozás nélkül.
- *Törlési probléma*: Törlés után – mivel csak egész sort törölhetünk -, elvesznek egyes adatok, információk, amelyekre a későbbiekben még szükség lenne.

A fenti – és más – anomáliák vizsgálata és megszüntetése leginkább a *függőségek vizsgálatával*, a *relációk felbontásával* és a *normálformákkal* lehetséges. Kezdő adatbázis-fejlesztő számára, alapvető relációalgebrai ismeretek hiányában, leginkább a függőségek (*funkcionális és többértékű kapcsolatok*) vizsgálhatók és a normálformák tűnnek használhatónak, és emellett alapvetően a redundancia kiszűrésére kell törekedni.

Funkcionális kapcsolatok

Funkcionális kapcsolatról akkor beszélünk, ha egy vagy több adat értékéből egy másik adat értéke következik.

Például a hallgató Neptun-kódja egyértelműen meghatározza a hallgató nevét:

```
Neptun-kód → Hallgató neve
```

A funkcionális kapcsolat vagy függőség baloldalát *meghatározónak* is nevezhetjük. A jobboldalon azonban csak egy darab érték állhat, tehát csak akkor beszélhetünk funkcionális függőségről, ha a meghatározó értékéből egy tulajdonságon belül csak egyetlen érték következik. Ha több, akkor nincs szó funkcionális kapcsolatról.

Például bár a névhez csak egyetlen születési év tartozhat, de mivel több embernek is lehet azonos neve, az adott névhez több születési év is hozzárendelhető.

Másik példánk a személyi szám és a gyerekek személyi számának kapcsolata. Mivel egy embernek több gyereke is lehet, itt sincs szó funkcionális függésről.

Az adatok közötti funkcionális függőség vizsgálata alapvetően tartalmi jellegű, az adatok „természetéből” következik.

A funkcionális függőség jobb oldalán egy tulajdonságból csak egyetlen érték állhat, de több tulajdonság is állhat, tehát egy tulajdonság több tulajdonságot is meghatározhat funkcionálisan.

Például az autó rendszámából egyértelműen következik az autó típusa és tulajdonosa:

```
Rendszám → Autó típusa, Autó tulajdonosa
```

Nemcsak a jobboldalon, hanem a baloldalon is több érték állhat.

Például a település neve és az utcanév meghatározza az irányítószámot:

```
Település neve, Utcanév → Irányítószám
```

Két tulajdonság *kölcsönös funkcionális függésben* is lehet egymással, ilyenkor a függés mindkét irányba igaz.

Például a többnejűséget kizáró országokban ilyen viszony van a házastársak személyi száma között:

```
Férj személyi száma → Feleség személyi száma,

Feleség személyi száma → Férj személyi száma
```

A funkcionális függőség egyik speciális formája a *teljes funkcionális függőség*.

Akkor beszélünk teljes funkcionális függőségről, ha a meghatározó oldalán nincs felesleges tulajdonság.

Például a következő funkcionális függőség nem teljes funkcionális függőség, mert az irányítószámot már az első két tulajdonság egyértelműen meghatározza:

```
Település neve, Utcanév, Házszám → Irányítószám
```

Többértékű függőség

Az adatok közötti kapcsolatok egy része nem fejezhető ki funkcionális függőséggel. Például egy hallgatói adatbázisban a *Hallgatókódból* egyértelműen következik a hallgató szakja, de a hallgatóhoz több szak is tartozhat, s ugyanarra a szakra természetesen több hallgató is jár. Ilyenkor egyik irányban sem lehet egyértelmű függőségről beszélni, mivel a meghatározó tulajdonság egyes adatértékeihez a meghatározott tulajdonság egy-egy értékhalmaza tartozik. Az ilyen függőséget *többértékű függőségnek* nevezzük. Előbbi példánknál maradva:

```
Hallgatókód →→ Szak
```

Többértékű függőségnél is előfordulhat, hogy a meghatározó értékéből több tulajdonság értéke következik:

```
Hallgatókód →→ Szak, Beiratkozás éve
```

A különböző függőségek ismeretében a korábban már említett általános anomália, a redundancia fogalma is jobban megragadható.

A redundancia

Ha valamely értéket vagy a többi adatból levezethető mennyiséget többszörösen tároljuk az adatbázisban, az adatbázisban redundancia van. A redundancia, a tárolóterület felesleges lefoglalása mellett, fölös adatbázis-frissítési és adatkarbantartási műveletekhez vezet, amelyek könnyen az adatbázis inkonzisztenciáját okozhatják. Az adatbázis akkor *inkonzisztens*, ha egymásnak ellentmondó tényeket tartalmaz.

Megjegyezzük, hogy a fizikai tervezés során az adatbázis-műveletek meggyorsítása érdekében esetenként redundáns tulajdonságokat is bevezethetünk.

A lenti táblában egyes adatokat többször is tárolunk:

DOLGOZÓK

Dolgozó	Osztálykód	Osztály neve	Osztályvezető	Beosztás
Kiss István	KÖ	Könyvelés	Molnár Katalin	könyvelő
Nagy József	PÜ	Raktár	Csiszár Zoltán	targoncavezető
Kovács Éva	KÖ	Könyvelés	Molnár Katalin	könyvelő

1. táblázat: Táblázat redundáns adatokkal

Az Osztály neve és az Osztályvezető oszlopok ebben a táblában való felvétele kettős hátránnyal jár:

- Ha az osztályvezető neve megváltozik, azt minden sorban módosítani kell.
- Ha új dolgozó kerül a táblába, nem elég csak az osztálykódot feltüntetni, az osztály nevét és az osztályvezető nevét is fel kell vennünk.

A redundancia azonban nem azonos az adatok többszörös tárolásával, sőt – mint ahogy később látni fogjuk – arra gyakran éppen a redundancia elkerüléséért van szükség.

TERMÉKEK

Termékkód	Terméktípus	Darabszám	Állapot	Listaár
KÖ-1	Könyv	6	új	2300 Ft
KÖ-2	Könyv	11	új	4800 Ft
KÖ-3	Könyv	1	használt	1290 Ft

2. táblázat: Adatok többszörös tárolása

A fenti táblában a terméktípusnál a többszörös előfordulás nem redundancia, hiszen a terméktípus az adott termék immanens tulajdonsága, feltüntetése nélkülözhetetlen. Ha azonban például Könyvek ÁFA-ja oszlop is lenne a táblázatban, az már redundancia lenne, hiszen az ÁFA funkcionálisan függ a termék típusától (esetünkben a könyvtől).

Redundanciával állunk szemben akkor is, ha a táblázatban olyan oszlopot tüntetünk fel, amely már valamilyen tárolt adatból egyértelműen levezethető adatokat tartalmaz.

A redundanciák kiszűrésének első lépése a függőségek feltárása lehet. A második pedig a normálformák alkalmazása.

Normálformák

A normálformák követése az esetek többségében kiküszöböli az anomáliákat, elsősorban a redundanciát.

Az adatbázis-kezelésben legalább öt normálformát használunk, de az első három alkalmazása általában tökéletesen elegendő.

Az első normálforma

Elterjedt (és a gyakorlatban tökéletesen használható), informális definíció szerint a tábla (a reláció) *első normálformában (1NF)* van, ha minden tulajdonsága egyszerű, nem összetett adatokat tartalmaz. Az egyszerű és az összetett adat szétválasztása mindig tartalmi vizsgálatot igényel. Általában arra kell törekedni, hogy az összetett adatokat a kezdetektől a lehető legkisebb egységekre bontva, külön oszlopokban tároljuk. A későbbiekben ugyanis sokkal nehezebb a már feltöltött adatbázisunk tábláinak egyes oszlopait szétválasztani. Az összetett adatok tárolása ráadásul az adatbázis-műveletek egy részét is nehézkessé teheti.

A formális definíció szerint azonban a tábla akkor van első normálformában, ha nincsenek benne *ismétlődő csoportok*.

Az igazán egzakt definícióhoz relációalgebrai levezetésre lenne szükség, de ezt elhagyjuk.

Megjegyezzük, hogy – érdekes módon – éppen az első normálformával kapcsolatban a mai napig elég éles elméleti vita folyik a szakirodalomban, s mivel a magyar nyelvű szakkönyvek többnyire ezeket a forrásokat követik, különböző forrásokban más magyarázatot találhatunk.

A fő probléma az „ismétlődő csoport” fogalmi megragadásában mutatkozik. A szigorú, coddi definíciót követő Elmashri - Navathe (2003) és a megengedő Date (2003) közötti elméleti vita számunkra nem különösebben fontos, de talán érdemes megjegyezni, hogy az általunk követett Date-féle definíció nem az egyetlen.

Date szerint egy tábla akkor van első normálformában, ha teljesíti a következő öt feltételt:

- Sorai között (részben vagy egészben) nincs sorrendi kapcsolat. (Véletlenül, vagy valamilyen nézetben megjelenhetnek sorba rendezve, de a sorrendnek adatbázis-kezelési szempontból nem lehet szerepe.)
- Oszlopai között (részben vagy egészben) nincs sorrendi kapcsolat. (Az értelmezés hasonló az előző ponthoz.)
- Nincsenek benne ismétlődő sorok, tehát nincs két, egymással egyező sor.

- Minden sor és oszlop metszete az adott értéktartomány egyetlen értékét tartalmazza.
- Minden oszlop szabályos, tehát a tárolt értéken túl nem tartalmazhatnak egyéb elemeket (azonosítókat, időbélyeget stb.).

Az első normálforma megsértésére általában az alábbiak utalnak:

- A táblában nincs elsődleges kulcs, s nem is lehet kialakítani. Ez könnyen azt jelezheti, hogy a táblában azonos sorok vannak.
- A tábla oszlopai vagy sorai között rendezettség tapasztalható.
- A táblában üres (null-) értékek találhatók. Az üres értékre ugyanis nem teljesül, hogy az adott értéktartomány része lenne. (Megjegyzés: Codd kései művei nem zárták ki a null értéket ezért ezt inkább vegyük „gyenge” szabálynak.)

Az alábbi tábla nincs első normálformában, a normalizálás végére viszont már abban jelenik meg:

VÁSÁRLÓK

Vásárlókód	Vezetéknév	Keresztnév	Telefonszám
23	Kiss	István	20-123-4567
34	Nagy	József	30-123-4567, 99-123-456
477	Kovács	Éva	70-123-4567

3. táblázat: Az első normálformát megsértő tábla

A tábla azért nincs első normálformában, mert a *Telefonszám* oszlop *Nagy József* két különböző telefonszámát is tartalmazza.

Adódik ilyenkor a lehetőség, hogy a telefonszámnak több oszlopot biztosítsunk.

VÁSÁRLÓK

Vásárlókód	Vezetéknév	Keresztnév	Telefonszám1	Telefonszám2
23	Kiss	István	20-123-4567	
34	Nagy	József	30-123-4567	99-123-456
477	Kovács	Éva	70-123-4567	

4. táblázat: Tábla üres értékekkel

A 4. táblázatban feltüntetett tábla azonban továbbra sincs első normálformában. Bár az még – egyes szerzők szerint – elfogadható lenne, hogy a táblában vannak üres értékek, de az semmiképpen sem, hogy a Telefonszám1 és a Telefonszám2 oszlop között rendezettség van.

Ha azt feltételezzük, hogy minden telefonszám csak egy vevőhöz tartozhat, a probléma megoldása során a táblát az alábbi szerkezetben két táblára kell bontanunk:

VÁSÁRLÓK NEVE

Vásárlókód	Vezetéknév	Keresztnév
23	Kiss	István
34	Nagy	József
477	Kovács	Éva

5. táblázat: Első normálformában lévő tábla

VÁSÁRLÓK TELEFONSZÁMA

Vásárlókód*	**Telefonszám**
23	20-123-4567
34	30-123-4567
477	70-123-4567
34	99-123-456

6. táblázat: Első normálformában lévő tábla

A normalizálás után létrejövő *Vásárlók neve* táblában a *Vásárlókód* oszlop, a *Vásárlók telefonszáma* táblában a *Telefonszám* oszlop elsődleges kulcsként funkcionál. A Vásárlókód a Vásárlók telefonszáma táblában idegen kulcsként funkcionál, ezért a két tábla között egy-a-többhöz kapcsolat van. Tehát egy vásárlóhoz több telefonszám tartozhat, de egy telefonszámhoz csak egy vásárlót lehet hozzárendelni. Az adatbázisséma tehát a következő:

`VÁSÁRLÓK NEVE (`**`Vásárlókód`**`, Vezetéknév, Keresztnév)`

`VÁSÁRLÓK TELEFONSZÁMA (`**`Telefonszám`**`, Vásárlókód*)`

A második normálforma

A a *második normálformának* (*2NF*) is több ismert megközelítése ismert, a legegyszerűbb definíció talán a következő:

Egy tábla (reláció) akkor van második normálformában, ha első normálformában van, s emellett a tábla minden, nem elsődleges kulcsként funkcionáló tulajdonsága teljes funkcionális függőségben van az elsődleges kulcs egészétől. A tábla (reláció) kulcsában nem szereplő tulajdonságokat (attribútumokat) *nem elsődleges tulajdonságnak* (attribútumnak) is nevezhetjük. Ha az elsődleges kulcs nem összetett kulcs (tehát egy oszlopból áll), akkor a tábla automatikusan második normálformában van, hiszen ekkor nem következhet be, hogy egy oszlop az elsődleges kulcs egy részétől függjön csak.

Az alábbi, összetett elsődleges kulcsot tartalmazó tábla nincs második normálformában:

VIZSGABEOSZTÁS

Tanterem	**Időpont**	Tárgy	Terem kapacitása
T1	8:00	Matematika	20
T2	10:00	Pénzügy	30
T1	12:00	Számvitel	20
T2	12:00	Statisztika	30
T1	16:00	Angol	20

7. táblázat: Tábla második normálformába rendezés előtt

A tábla tulajdonságai között két teljes funkcionális függés írható le:

`Tanterem, Időpont → Tárgy`

`Tanterem → Terem kapacitása`

A *Terem kapacitása* tehát csak az elsődleges kulcs egy részéhez képest van teljes funkcionális függésben, ezért nincs második normálformában. Ha azonban a táblát az alábbi módon, két részre bontjuk, mindkét tábla második normálformába kerül:

VIZSGABEOSZTÁS

Tanterem	**Időpont**	Tárgy
T1	8:00	Matematika
T2	10:00	Pénzügy
T1	12:00	Számvitel
T2	12:00	Statisztika
T1	16:00	Angol

8. táblázat: Tábla második normálformában

TERMEK

Tanterem	Terem kapacitása
T1	20
T2	30

9. táblázat: Tábla második normálformában

Mindkét tábla második normálformában van, mert a nem elsődleges kulcsként funkcionáló (nem elsődleges) tulajdonságok teljes funkcionális függésben vannak az elsődleges kulcs egészével:

`Tanterem, Időpont → Tárgy`

`Tanterem → Terem kapacitása`

A két táblából álló adatbázis sémája tehát a következő:

`VIZSGABEOSZTÁS (`**`Tanterem*, Időpont,`** `Tárgy)`

`TERMEK (`**`Tanterem*,`** `Terem kapacitása)`

Első ránézésre úgy tűnhet, hogy a két tábla között egy-az-egyhez kapcsolat van, de ez nincs így, hiszen a *Tanterem* oszlop önmagában csak a *Termek* táblában elsődleges kulcs, a másik táblában az Időponttal együtt alkot elsődleges kulcsot. A két tábla között valójában – mivel a Termek tábla elsődleges kulcsa a másik táblában idegen kulcs – egy-a-többhöz kapcsolat van.

A harmadik normálforma

A harmadik normálformára (*3NF*) is több definíciót találhatunk a különböző forrásokban, bár ezek természetesen – az előzőekhez hasonlóan – egymással lényegében ekvivalensek.

Az általunk leghelyesebbnek tartott definíció a következő:

Egy tábla (reláció) akkor van harmadik normálformában, ha második normálformában van, és emellett a nem elsődleges kulcsként funkcionáló (nem elsődleges) tulajdonságok között nincs funkcionális függés.

A következő tábla második normálformában van, de nincs harmadik normálformában:

VIZSGABIZOTTSÁGOK

Vizsga	Vizsgaelnök	Elnök személyi száma
Matematika	Kis István	1-671107-1234
Pénzügy	Molnár Attila	1-600404-2345
Számvitel	Kovács Éva	2-650501-3456
Statisztika	Molnár Attila	1-600404-2345

Angol	Szabó Ferenc	1-751202-7891

10. táblázat: Tábla harmadik normálformába rendezés előtt

A fenti táblában a vizsgaelnök személyének ismétlődése nem redundancia, de a vele funkcionális függésben lévő adatok ismétlődése már redundanciához vezet:

`Vizsga → Vizsgaelnök → Elnök személyi száma`

A redundanciához vezető funkcionális függést az alábbi táblaszerkezettel lehet feloldani:

VIZSGABIZOTTSÁGOK

Vizsga	Vizsgaelnök
Matematika	Kis István
Pénzügy	Molnár Attila
Számvitel	Kovács Éva
Statisztika	Molnár Attila
Angol	Szabó Ferenc

11. táblázat: Tábla harmadik normálformában

VIZSGAELNÖKÖK

Vizsgaelnök	Elnök személyi száma
Kis István	1-671107-1234
Molnár Attila	1-600404-2345
Kovács Éva	2-650501-3456
Szabó Ferenc	1-751202-7891

12. táblázat: Tábla harmadik normálformában

A fenti táblákban a nem elsődleges tulajdonságok között már nincs funkcionális függés, a nem elsődleges azonosítók csak az elsődleges azonosítóktól (a kulcs részeitől vagy – nem összetett kulcsok esetén – a kulcstól) függnek funkcionálisan:

`Vizsga → Vizsgaelnök`

`Vizsgaelnök → Elnök személyi száma`

A táblaszerkezet alapján látható, hogy a két tábla között egy-a-többhöz kapcsolat van:

`VIZSGABIZOTTSÁGOK (`**`Vizsga,`** `Vizsgaelnök*)`

`VIZSGAELNÖKÖK (`**`Vizsgaelnök,`** `Elnök személyi száma)`

Az adatbázis-kezelés során az első három normálforma általában elegendő az adatbázistáblák vizsgálata, kialakítása során. Létezik ugyan még néhány normálforma (Boyce-Codd, negyedik és ötödik normálforma), amelyek speciális esetekben további redundanciák kiküszöbölésében lehetnek a segítségünkre, de ezek ismerete nélkül is jól működő és minimális redundanciával terhelt adatbázisokat alakíthatunk ki.

Az adatbázis-tervezés lépései

Az alábbiakban bemutatjuk az adatbázis-tervezés folyamatának egyik lehetséges, nyolc lépésből álló módszerét.

Követelmények elemzése

Először mindig tisztázni kell az adatbázis létrehozásának pontos célját. Érdemes mindig megvizsgálni a már kész megoldásokat, sok esetben jó kiindulópontot jelenthetnek. Készítsünk interjút az adatbázis felhasználóival, mérjük fel, hogy milyen adatokra számíthatnak, hogyan dolgozzák fel azokat, gyűjtsük össze az adatfelvételre használt űrlapokat! Foglaljuk össze, hogy az adatbázisból milyen információkat szeretnénk kinyerni! Végül írjuk össze, hogy milyen témákról, mely egyedekről, milyen adatokat szükséges tárolni!

Táblák meghatározása

Az összegyűjtött adatokból lehatároljuk az egyedet, majd az egyed tulajdonságait táblákba rendezzük.

Tulajdonságok, mezők meghatározása

A táblákat felépítő oszlopok (tulajdonságok, mezők) meghatározásával konkrétan, tulajdonságaikon keresztül definiáljuk az egyes egyedtípusokat.

Elsődleges kulcs meghatározása

A táblák közötti kapcsolatok biztosításhoz meghatározzuk a táblák elsődleges kulcsait.

Kapcsolatok meghatározása

Az elsődleges kulcsok és az idegen kulcsok alapján meghatározzuk a táblák közötti kapcsolatokat.

Normalizálás

A normálformák segítségével kiszűrjük a redundanciával fenyegető táblaszerkezeteket, s – ha szükséges – a táblák átalakításával vagy új táblák létrehozásával normalizáljuk az adatbázis tábláit.

Ellenőrzés

Az ellenőrzés során nagyon alapos munkára kell törekedni, mert a későbbiekben a változtatás esetenként jelentős akadályokba ütközhet.

További objektumok tervezése

Az első hét lépés után érdemes az adatbázis-kezelővel (esetünkben az Access-szel) elkészíteni az adatbázis tábláit és definiálni kulcsait, kapcsolatait. Miután ezzel végeztünk, felmerülhet az igény, hogy az adatbázis-kezelő által biztosított objektumok közül is létrehozzunk és testre szabjunk néhányat.

Adatbázis-kezelés Microsoft Access-szel

Az adatbázis-kezelők meghatározó része a logikai adatbázisok kialakítására és kezelésére alkalmas rendszer, azonban a legtöbb adatbázis-kezelő számos olyan egyéb objektumot biztosít, amelyek segítségével az adatbázist – a fizikai szint ismerete nélkül – kifinomult, felhasználóbarát adatbázis-alkalmazássá fejleszthetjük.

Az Access kisebb adatbázisok számára optimális és rendkívül felhasználóbarát szoftver.

Az általunk használt 2010-es verzió legfontosabb újítása a korábbiakhoz képest, hogy könnyen létrehozhatunk webes adatbázisokat, s azokat közzétehetjük egy SharePoint-webhelyen (sajnos a tárhelyszolgáltatók jelentős része nem támogatja). Emellett az felhasználói felület is jelentősen megújult, új adattípusokat is használhatunk.

A jegyzet célja azonban nem elsősorban az Access, hanem az adatbázis-kezelés gyakorlatias alapjainak bemutatása. Éppen ezért az Access minden részletre kiterjedő bemutatását elhagyjuk, s inkább csak a – konkrét feladatokhoz kapcsolódó – legfontosabb elemekre koncentrálunk.

Bevezető feladatsor

Hozzuk létre – a 9. ábrán adataival együtt felírt – *Államvizsga* nevű adatbázist:

VIZSGABIZOTTSÁGOK (**Vizsga**, Vizsgaelnök*)

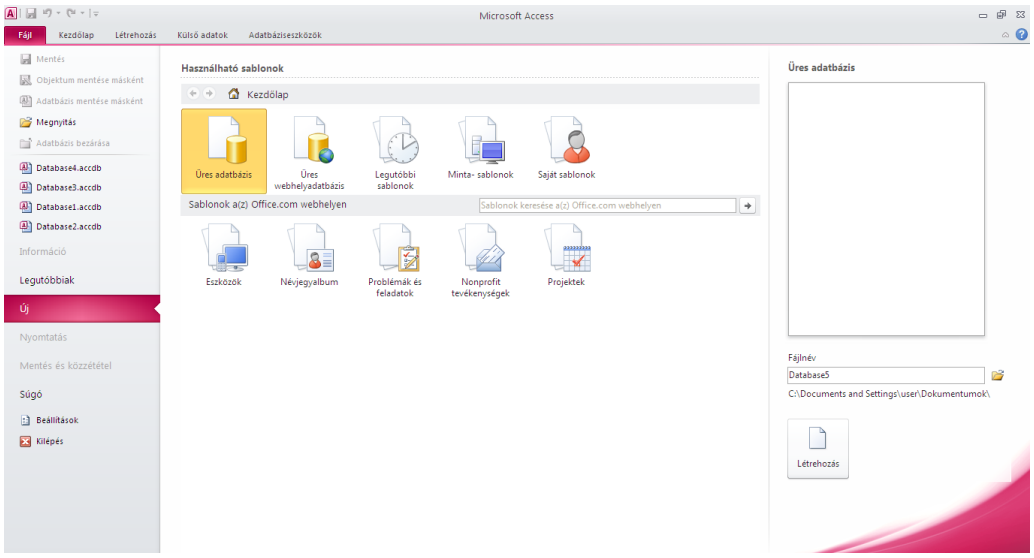
VIZSGAELNÖKÖK (**Vizsgaelnök**, Elnök személyi száma)

Az egyes mezőkhöz tartozó adattípusok:

- *Vizsga*: szöveges mező
- *Vizsgaelnök*: szöveges mező
- *Elnök személyi száma*: szöveges mező

Adatbázis-kezelő megnyitása

Első lépésben nyissuk meg az adatbázis-kezelőnket! Ez legegyszerűbben úgy történhet, hogy az Asztalon elhelyezkedő ikonra kattintunk, de ha nincs ilyen, akkor Microsoft XP operációs rendszerben a *Start menü/Minden program/Microsoft Office/Microsoft Access 2010* pontokra kattintva nyithatjuk meg a szoftvert. Megnyitás után az 1. ábrán látható felületet kapjuk.



1. ábra: Az Access nyitóoldala

Adatbázis definiálása

A nyitóoldalon (a *Fájl* lapon) megnyithatunk egy már létező adatbázist, ill. az új adatbázisunkhoz a gépünkön vagy az Office.com webhelyen lévő sablonokat is megnyithatjuk, de nekünk most egy új adatbázist kell létrehoznunk. Az ablak jobb alsó részében válasszuk ki a mentési helyet, majd adjunk nevet a fájlnak! Mi az *Államvizsga* fájlnevet választottuk.

Megjegyezzük, hogy elvileg semmi sem tiltja, hogy a különböző elemeknek, objektumoknak a magyar írásmódot követve, ékezetes betűket használva adjunk nevet. Azonban tapasztalataink szerint –

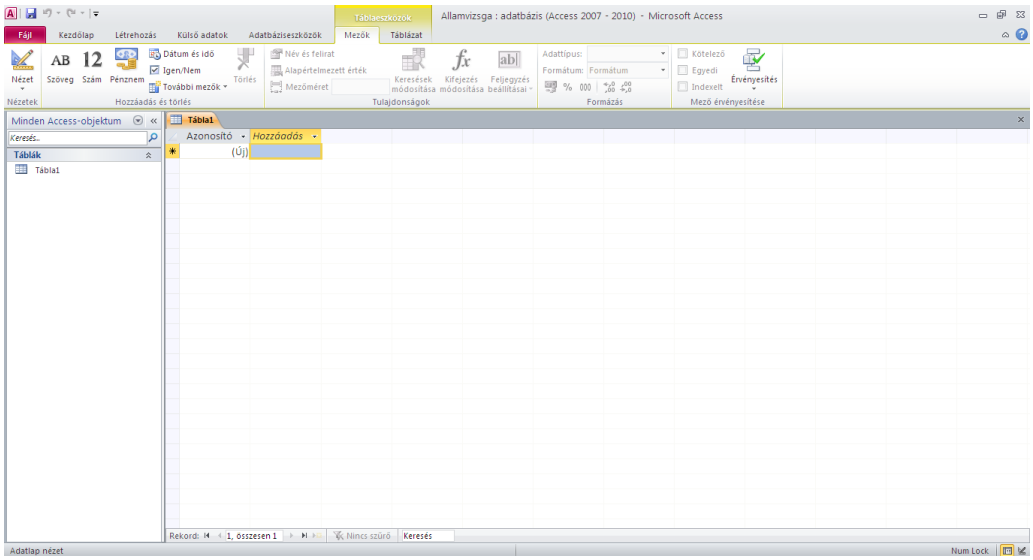
különösen webes alkalmazásoknál – az objektumok neveinél érdemes az ASCII-karakterek (az angol ábécé betűi, számok és néhány egyéb írásjel) közül válogatni, ezzel számos karakterkódolási problémát elkerülhetünk. Természetesen arra kell törekedni, hogy a végfelhasználó előtt megjelenő feliratokat mindig a magyar karakterkészletből állítsuk össze.

Mi a jegyzetben első felében az objektumneveknél – ha szükséges – ékezetes betűket is használunk, azonban ezt „élesben” érdemes inkább elkerülni, ezért később elhagyjuk az ékezetes karaktereket.

Táblatervezés

Miután a mentés helyét kiválasztottuk és beírtuk a fájlnevet, kattintsunk a *Létrehozás* gombra. Az adatbázis-kezelő ezután létrehozza az adatbázist és megjelenít egy *Tábla1* nevű alapértelmezett táblát.

A *Táblaeszközök* lapcsoport alatt a *Mezők* és *Táblázat* lapok közül választhatjuk ki a különböző táblaműveleteket.



2. ábra: A Táblaeszközök lapcsoport Adatlap nézetben

A képernyő bal szélén elhelyezkedő *Navigációs ablakban* jelenleg a Tábla1 nevű tábla látható, amely valójában csak „virtuálisan” létezik, hiszen semmilyen tulajdonságát, egyetlen sorát vagy oszlopát sem definiáltuk. Kattintsunk jobb egérkapcsolóval a Tábla1 táblát jelző ikonra, majd a megjelenő helyi menüből válasszuk ki az *Tervező nézet* pontot!

A Tervező nézetet a Kezdőlap lap Nézet pontja alatt lenyíló listából is kiválaszthatjuk. A Tervező nézet mellett számunkra még az *Adat* nézet fontos, hiszen a tábla adatait ebben a nézetben jeleníthetjük meg.

Ha így teszünk, felugrik a *Mentés másként* párbeszédpanel. A *Táblanév:* alá írjuk be a *Vizsgaelnökök* táblanevet!

A Tervező nézetben megjelenő tábla Vizsgaelnökök táblafüle alatt a *Mezőnév* és az *Adattípus* oszlopokat látjuk.

A Mezőnévnél egymás alá vegyük fel a következő értékeket: *Vizsgaelnök, Elnök személyi száma*

A szóközök használata a mezőnevekben nem túl szerencsés, számos adatbázis-kezelő ma sem támogatja. A mezőnevekben a szóközöket érdemes akkor is „_” jellel helyettesíteni, ha ez nem kötelező (az Accessben nem az), mivel a kettős szóközleütésből fakadó hibákat „_” jel használatával jó eséllyel kiküszöbölhetjük. Mi azonban a jegyzetben esztétikai okokból többnyire nem így járunk el.

Adattípusok

Miután felvettük a két mezőnevet, az Adattípusnál vegyünk fel a Vizsgaelnök és az Elnök személyi száma oszlopokhoz *Szöveg* adattípust! Az adattípust a cellára kattintva, majd a megfelelő értéket a megjelenő lenyíló listából kiválasztva lehet felvenni.

Mivel az adattípusok beállítása kitüntetett szereppel bír, érdemes ezzel kicsit bővebben foglalkozni.

Az adattípus a mező egyik legfontosabb tulajdonsága, hiszen az adattípus meghatározza, hogy az adott oszlopban milyen típusú és „méretű" adatokat tárolunk, azokkal milyen műveleteket végezhetünk. A különböző adattípusok meghatározzák egy-egy adat tárhelyigényét is. A különböző adattípusok tárhelyigénye között nagyon nagy különbség lehet, ezért – különösen nagy adatbázisoknál – arra kell törekedni, hogy olyan adattípust válasszunk, amely lehetővé teszi az oszlophoz tartozó összes adat tárolását és az összes művelet elvégzését, de emellett a lehető legkisebb tárterületet veszi igénybe.

Az adattípusok némileg megtévesztőek lehetnek, hiszen például *Szöveg* adattípusnál az oszlopban szövegeket és számokat is tárolhatunk, de például *Szám* adattípusnál kizárólag számokat. Éppen ezért a legfontosabb adattípusokkal érdemes megismerkedni.

ALAPTÍPUSOK

Adattípus	**Megjeleníthető adatok**
Szöveg	Viszonylag rövid alfanumerikus értékek
Szám	Számértékek
Pénznem	Pénzben tárolt adatok
Igen/Nem	Igen vagy nem értékek, tehát amelyek egy logikai kifejezés igaz vagy hamis visszatérési értékeiként értelmezhetők
Dátum és idő	Dátum és időértékek 100-tól a 9999-es évig
Rich Text	Színekkel és betűtípus-formátumokkal formázható számok és betűk
Számított mező	Az adott táblában található más mezőkre hivatkozó számítási eredmény
Melléklet	Adatbázissorokhoz kapcsolódó különböző formátumú csatolt fájlok
Hivatkozás	Szövegként tárolt és hivatkozáscímként használt szöveg, vagy betűk és számok kombinációja
Feljegyzés	Hosszabb alfanumerikus értékek
Keresés és kapcsolat	Táblából vagy lekérdezésből beolvassa az értéklistát, vagy a mező létrehozása során már definiált, statikus értékkészletet jelenít meg. A típust kiválasztva a *Keresés* varázsló is elindul.

13. táblázat: Alapvető adattípusok az Accessben

Egyes adattípusok altípusainak ismerete azonban szintén fontos lehet a tervezés során, ezért a későbbiekben érdemes megismerkedni a legfontosabb altípusokkal is.

A Tervező nézet bal alsó sarkában található *Mezőtulajdonságok* fül alatt számos mezőtulajdonságot beállíthatunk. Az adattípus jelentős mértékben meghatározza a beállítható mezőtulajdonságokat. A Szöveg típusú mezők mérete pl. a *Mezőméret* tulajdonsággal szabályozható. *Szám* és *Pénznem* típusú mezőknél a Mezőméret határozza meg a felvehető értékek tartományát.

Mivel a Szám adattípushoz tartozó Mezőméret értékeit – lényegében a Szám adattípus altípusait – elég gyakran módosítjuk, érdemes ezekkel részletesebben is megismerkedni.

A Szám adattípuson belül a Mezőméretnél az alábbi lehetőségek közül választhatunk:

- **Bájt** – 0 és 255 közötti egész számok tárolását teszi lehetővé, az ilyen mezőméretű adat 1 bájtnyi tárterületet foglal el.

- **Egész** – -32 768 és 32 767 közötti egész számokhoz választható. Az adat 2 bájtnyi tárterületet foglal el.
- **Hosszú egész** – -2 147 483 648 és 2 147 483 647 közötti egész számokat tárolhatunk a segítségével. 4 bájtnyi tárterületet foglal el.
- **Egyszeres** – -3,4 x 1038 és 3,4 x 1038 közötti, hét számjeggyel megadott lebegőpontos számértékekhez választható. 4 bájtnyi tárterületet foglal el.
- **Dupla** – -1,797 x 10308 és 1,797 x 10308 közötti, tizenöt számjeggyel megadott lebegőpontos számértékekhez. 8 bájtnyi tárterületet foglal el.
- **Replikációs azonosító** – A replikációhoz szükséges. 16 bájtnyi tárterületet foglal el. (A replikáció során az Access-adatbázisról több speciális másolat (replika) készül. Számunka az adattípus egyelőre nem fontos.)
- **Decimális** – -9.999... x 1027 és 9.999... x 1027 közötti számértékekhez használható. 12 bájtnyi tárterületet foglal el.

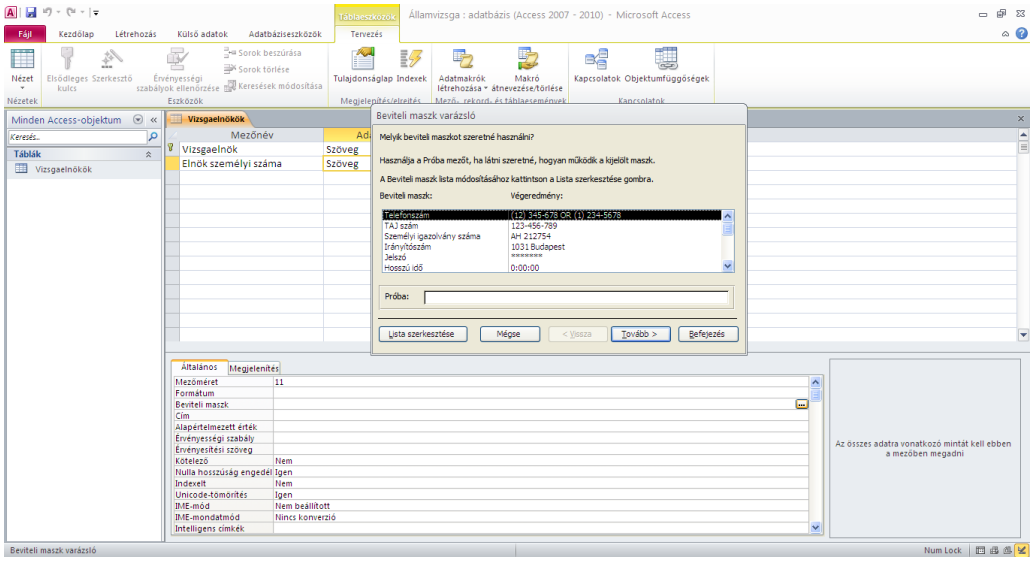
A Vizsgaelnök mező mezőméretét csökkentsük le 255-ről 50-re, az értéket numerikusan az Általános/Mezőméret cellájába beírva! (Hiszen nem valószínű, hogy 50 karakternél hosszabb neveket kellene tárolnunk.)

Az Elnök személyi száma oszlopot szintén érdemes lecsökkenteni, mégpedig 11 karakterre, hiszen a személyi szám 11 számból áll. Az olvasóban talán felmerülhet, hogy a személyi számot összekötő karaktereket (kötőjeleket) is tárolni kellene, azonban elegánsabb és célravezetőbb megoldást is alkalmazhatunk.

Beviteli maszk

Kattintsunk bele bal egérkapcsolóval az *Általános*/*Beviteli* maszk cellába!

Miután belekattintottunk, a cellától jobbra megjelenik egy három pontot tartalmazó vezérlőelem. Kattintsunk rá! Az adatbázis-kezelő ezután rákérdez, hogy el akarjuk-e menteni a táblát. Mentsük el! Mentés után megjelenik a *Beviteli maszk varázsló*.



3. ábra: A Beviteli maszk varázsló megjelenítése

Beviteli maszk segítségével hatékonyan tudjuk szabályozni, hogy az egyes mezőkbe milyen formátumban vigyük be az adatokat. A 14. táblázatban a helyőrzőként és konstansként használható karakterek láthatók.

A BEVITELI MASZK KARAKTEREI

Karakter	Leírás

0	Kötelezően beírandó számjegy (0-9).
9	Tetszőlegesen beírható számjegy (0-9).
#	Tetszőlegesen beírható számjegy, szóköz, pluszjel, mínuszjel. Ha a felhasználó nem ad meg semmit, az Access szóközt szúr be.
L	Kötelezően beírandó betű.
?	Tetszőlegesen beírható betű.
A	Kötelezően beírandó betű vagy számjegy.
a	Tetszőlegesen beírható betű vagy számjegy.
&	Kötelezően beírandó karakter vagy szóköz.
C	Tetszőlegesen beírható karakter vagy szóköz.
. : , ; - /	A Windows területi beállításaitól függő ezres, tizedes, dátum- és időelválasztók.
>	A következő karakterek nagybetűként jelennek meg.
<	A következő karakterek kisbetűként jelennek meg.
!	A beviteli maszk nem jobbról balra, hanem balról jobbra tölti fel az adatokat.
\	A következő karakterek betűhíven jelennek meg.
""	Az idézőjelek közötti karakterek betűhíven jelennek meg.
Többi karakter (Pl. kötőjel.)	Általában elválasztó elemként használható.

14. táblázat: A beviteli maszk karakterei

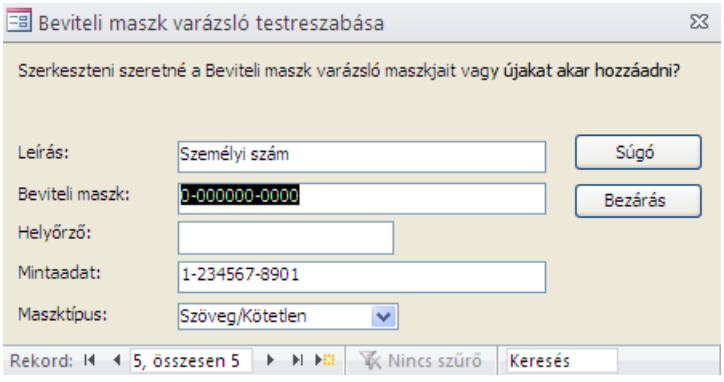
A fenti táblázat alapján a személyi szám beviteli maszkja a következő karakterekkel állíthatjuk be:

```
0-000000-0000
```

A „0” a kötelezően beírandó számjegyeket, a „-” pedig az elválasztó karaktert jelöli.

Miután a Beviteli maszk varázslót megjelenítettük, kattintsunk a *Lista szerkesztése* gombra, majd – mivel a személyi számnak megfelelő maszk nincs a listában – a felugró *Beviteli maszk varázsló testreszabása* ablak láblécében az *Új (üres) rekord* vezérlőre!

A személyi számhoz tartozó beviteli maszk beállítását a 4. ábra mutatja.



4. ábra: A beviteli maszk beállítása

A felvett beviteli maszkot a *Tovább/Befejezés* gombra kattintva vihetjük be a mezőtulajdonságok közé. A bevitt érték több kiegészítő elemet is tartalmaz, de az általunk bevitt 0-000000-0000 tökéletesen elegendő, és ekvivalens a megjelenő *0\-000000\-0000;;* maszkkal.

Elsődleges kulcs beállítása

Az Access 2010-es változata automatikusan elsődleges kulcsot rendel az első mezőhöz, ezt a mezőnévtől balra elhelyezkedő kulcs ikon mutatja.

Ha más oszlophoz vagy oszlopokhoz szeretnénk elsődleges kulcsot rendelni, akkor

- vagy a *Táblaeszközök/Tervezés/Elsődleges kulcs* vezérlőre kattintunk,
- vagy jobb egérkapcsolóval a mezőnévre kattintunk, és a megjelenő helyi menüből az *Elsődleges kulcs* menüpontot választjuk.

Új tábla felvétele

Miután az első táblánkban mindent beállítottunk, kattintsunk a Vizsgaelnökök fülre jobb egérkapcsolóval, s a megjelentő helyi menüben válasszuk ki a Mentés menüpontot.

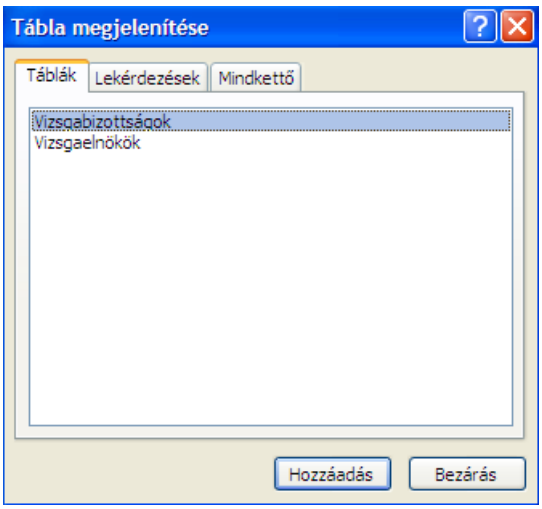
Hozzuk létre a második táblánkat a *Létrehozás/Tábla vezérlőre* kattintva! A táblát ezután a tanult módon jelenítsük meg Tervező nézetben, s mentsük el Vizsgabizottságok néven!

A tábla felvétele egyszerű, hiszen a Mezőnév alatt csak a Vizsga és a Vizsgabizottságok értéket kell felvennünk, az Adattípushoz mindkét esetben Szöveg kerül, és az automatikusan létrehozott elsődleges kulcs is megfelelő. Ha mindent beállítottunk, mentsük el a táblát!

Táblák összekapcsolása

Már korábban arra a következtetésre jutottunk, hogy az elsődleges kulcs és az idegen kulcs viszonya alapján a két tábla között egy-a-többhöz kapcsolat van. Elvben nem lenne szükség a kapcsolat külön beállítására a két tábla között, azonban kezdő felhasználók helyesen teszik, ha az Access által biztosított eszközökkel mindig beállítják a táblák közötti kapcsolatokat.

Két tábla között általában a *Táblázat/Kapcsolatok* vezérlőre kattintva célszerű beállítani a táblák közötti kapcsolatot. Kattintsunk rá a jelzett vezérlőre! Kattintás után megjelenik a *Tábla megjelenítése* párbeszédpanel.

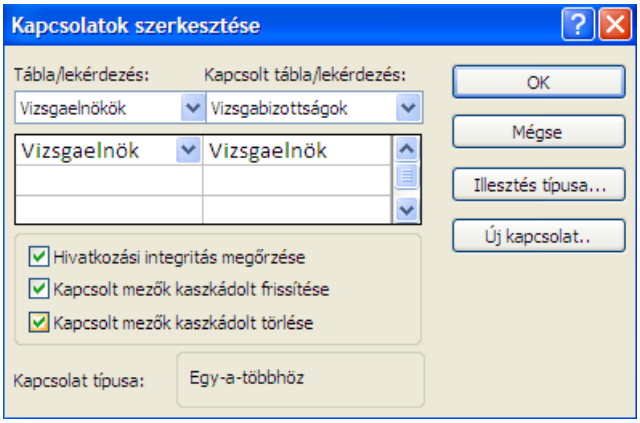


5. ábra: Kapcsolatok - Tábla megjelenítése

A táblákat egyenként válasszuk ki, és a *Hozzáadás* gombra kattintva adjuk őket hozzá a Kapcsolatok panel tervezőfelületéhez!

A tervezőfelületen jelöljük ki a kapcsolatot hordozó elsődleges vagy idegen kulcsot, tartsuk lenyomva fölötte a bal egérkapcsolót, majd a kapcsológombot folyamatosan lenyomva, „fogd és vidd” (drag and drop) technikával vigyük át a másik tábla kapcsolódó mezőneve fölé, majd engedjük el az egérkapcsolót!

Ha jól dolgoztunk, a felugró *Kapcsolatok szerkesztése* párbeszédpanel a 6. ábrán látható szerkezetet követi.



6. ábra: Kapcsolatok szerkesztése

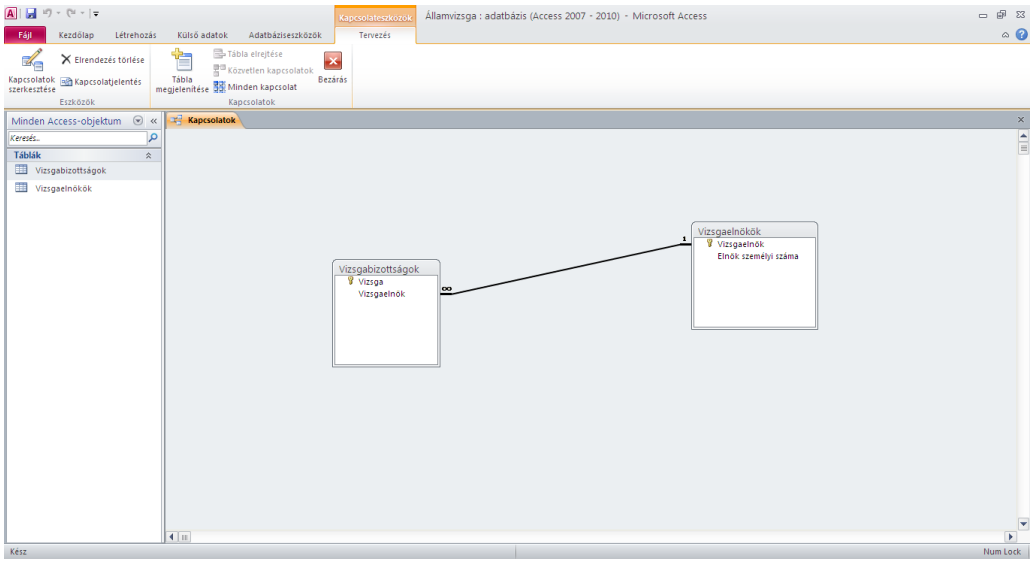
A kapcsolatok szerkesztése párbeszédpanelen beállítható a hivatkozási integritás néhány paramétere.

A hivatkozási integritás biztosításával elsősorban az a célunk, hogy ne jöjjenek létre *árvarekordok*, amelyek nem létező sorokra hivatkoznak. Ha a hivatkozási integritás megőrzését engedélyezzük, az adatbázis-kezelő minden hivatkozási integritást veszélyeztető műveletet megakadályoz.

Hivatkozási integritás megőrzésének engedélyezése után a következő szabályok lépnek életbe:

- Árvarekordokhoz vezetne, ha a kapcsolódó tábla idegen kulcs mezőjébe olyan értéket írnánk, amely nem létezik az elsődleges tábla elsődleges kulcsot alkotó mezőjében, ezért az idegen kulcs értékei csak a hozzá kapcsolódó elsődleges kulcs értékei közül kerülhetnek ki.
- Az elsődleges táblában nem lehet olyan sort törölni, amelyekhez a kapcsolódó táblában sorok tartoznak. Azonban a *Kapcsolt mezők kaszkádolt törlése* jelölőnégyzet kiválasztásával biztosítani tudjuk ezt a funkciót. Ekkor az elsődleges kulcshoz tartozó érték törlésekor az adatbázis-kezelő az összes hozzá kapcsolódó sort is törli.
- Az elsődleges táblában nem lehet olyan elsődleges kulcshoz tartozó értéket módosítani, amelyekhez a kapcsolódó táblában sorok tartoznak. Azonban a *Kapcsolt mezők kaszkádolt frissítése* jelölőnégyzet kiválasztásával biztosítani tudjuk ezt a funkciót. Ekkor az elsődleges kulcshoz tartozó érték módosításakor az adatbázis-kezelő az összes hozzá kapcsolódó sor idegen kulcsának értékét is módosítja.

Esetünkben csak olyan vizsgaelnököt akarunk a vizsgákhoz rendelni, aki szerepel a Vizsgaelnökök táblában, s a kapcsolt mezők törlését és frissítését is biztosítani szeretnénk. Ezért mindhárom jelölőnégyzetbe kattintsunk bele, majd kattintsunk az OK gombra! A művelet után vizuálisan is kirajzolódik az adatbázistáblák közötti kapcsolat.



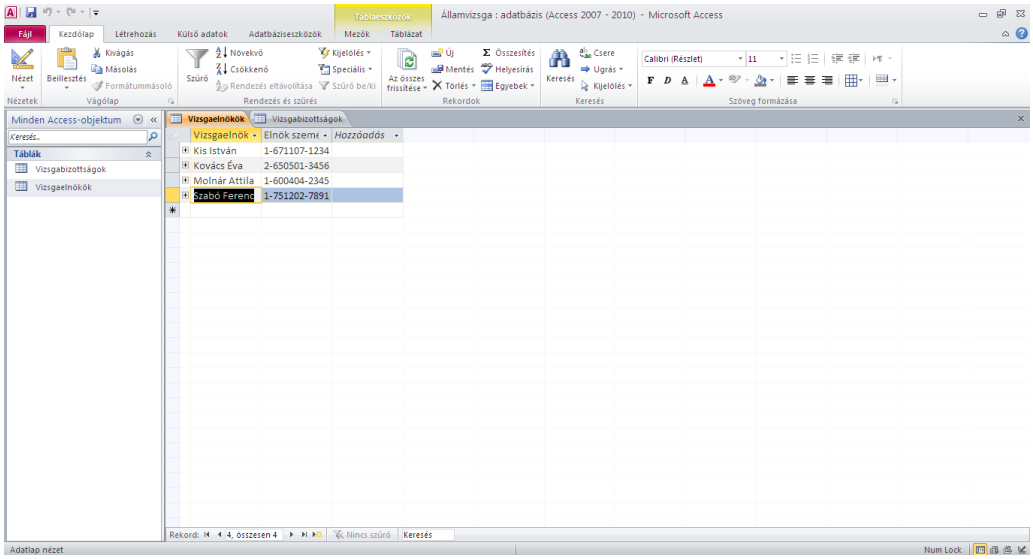
7. ábra: Kapcsolatok nézet

A Kapcsolatok fülre jobb egérkapcsolóval kattintsunk rá, s a megjelenő helyi menüben először mentsük el, majd zárjuk be a Kapcsolatok nézetet.

Adatfeltöltés

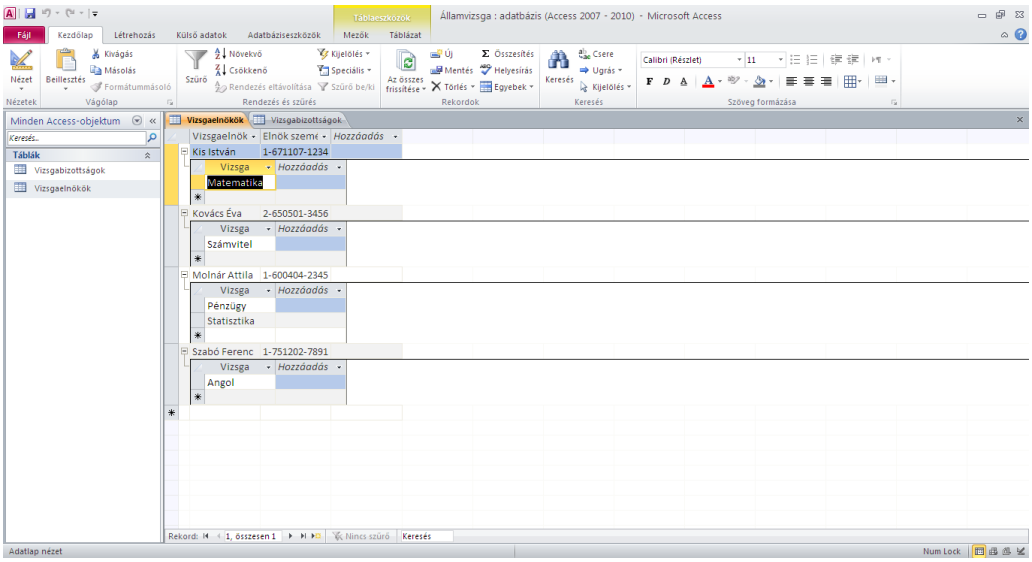
Bár „élesben” az adatfeltöltés általában felhasználóbarát űrlapokkal történik, a táblákat Adatlap nézetben is feltölthetjük adatokkal. A Navigációs sávban kattintsunk duplán (vagy jobb egérklikk után a helyi menüből válasszuk ki a Megnyitást, vagy a tábla kijelölése után a Kezdőlap lap Nézet lenyíló listájából válasszuk ki az Adatlap nézetet), s töltsük fel a táblákat adatokkal! (Lásd a 11. és a 12. táblázat adatait!)

Az adatfeltöltést a hivatkozási integritásnál írtak alapján kezdjük a Vizsgaelnökök táblánál!



8. ábra: Tábla Adatlap nézetben

Az első tábla feltöltése után választhatjuk azt a módszert is, hogy nem jelenítjük meg a kapcsolódó táblát, hanem az elsődleges tábla soraitól balra elhelyezkedő kis keresztekre kattintva megjelenítjük a *segédtáblát*, s a kapcsolódó tábla értékeit annak megnyitása nélkül visszük fel. (A másik táblában csak mentés után jelennek meg az adatok.)



9. ábra: Tábla Adatlap nézetben segédtáblával

A Hozzáadás lista megjelenítése után Adatlap nézetben is új oszlopokat adhatunk a táblához.

Adatfeltöltés után mentsük el a módosításokat, majd zárjuk be a tábla Adatlap nézetét.

Táblaszerkezet módosítása

A táblákhoz természetesen – a korábban vázolt szabályok betartása mellett – később is hozzáadhatunk új oszlopot.

Egészítsük ki a Vizsgabizottságok táblát egy Időpont oszloppal!

Nyissuk meg a tábla Tervező nézetét, vegyük fel az új oszlopot, a hozzá tartozó adattípus Dátum/Idő legyen, s a Formátum mezőtulajdonságot állítsuk be *Általános dátumra*!

Tegyük fel, hogy az összes záróvizsga időpontja 2012. február 20-ra esik, csak a kezdési időpont különbözik! Ekkor felesleges lenne állandóan ugyanazokat a karaktereket begépelni. Az Időpont oszlop Alapértelmezett érték cellájába ezért vigyük be a következő értéket:

```
#2013.02.20. 8:00:00#
```

Ha konkrét dátumértékekkel dolgozunk, a dátumérték elé és mögé „#” jelet kell beírnunk. Enélkül az adatbázis-kezelő a bevitt értéket hibásnak tekintené, mivel a dátumértékekben lévő, elválasztó pont a kiterjesztéseknél és az adatbázis-hivatkozásoknál más célt szolgál.

A bevitt alapértelmezett érték azonban csak az érték definiálása után felvett rekordokra érvényes, s ott sem kötelező a megtartása, bármikor módosítható. Mivel mi egyelőre új tantárgyat nem akarunk felvenni, valójában felesleges volt az Alapértelmezett érték definiálása.

Az Alapértelmezett értéktől eltérően az Érvényességi szabály kötelező jellegű és utólag is érvényesíthető a szabály felvétele előtt táblába írt rekordokra.

Tegyük fel, hogy biztosítani akarjuk, hogy az épp aktuális napnál korábbi érték semmiképpen se kerüljön a táblába, hiszen ez minden bizonnyal elírás lenne.

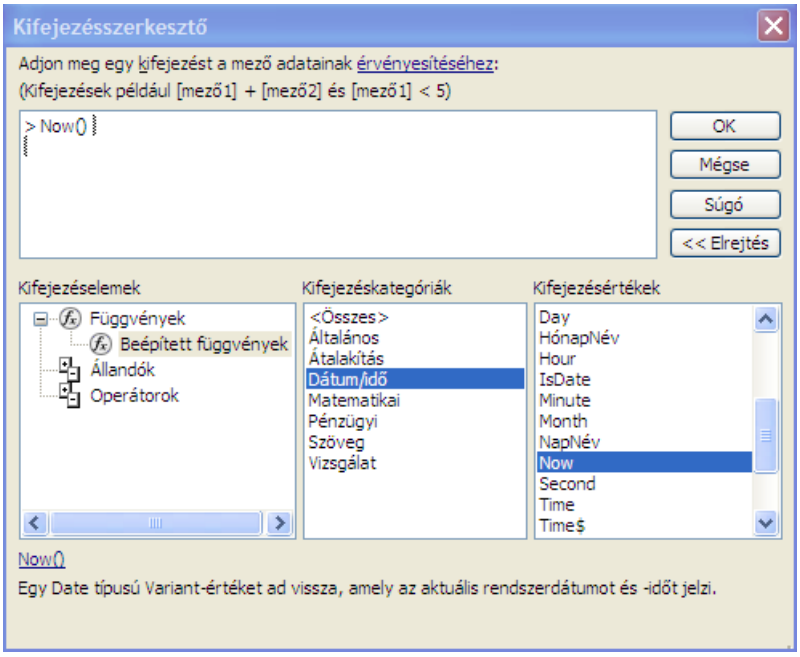
Kattintsunk bele az *Érvényességi szabály* cellába! Miután belekattintottunk, a cellától jobbra megjelenik egy három pontot tartalmazó vezérlő. Kattintsunk rá!

Kattintás után megjelenik a *Kifejezésszerkesztő* párbeszédpanel, amelyet alapvetően bonyolultabb kifejezéseknél érdemes használni.

A kifejezések függvényeket, operátorokat, állandókat és azonosítókat (oszlopok, táblák, űrlapok és lekérdezések nevei) tartalmazhatnak. A Kifejezésszerkesztő segítségével az összes ilyen összetevőt megkereshetjük és beszúrhatjuk a kifejezésünkbe.

A Kifejezésszerkesztő legfontosabb részei a következők:

- *Útmutatás és súgóhivatkozás*: Az ablak felső sávjában, attól függően, hogy a kifejezést milyen célból szerkesztjük, alapvető információkat látunk a környezetről.
- *Kifejezésmező*: Közvetlenül az előző rész alatt vehetjük fel a kifejezést beírással vagy kifejezéselemek hozzáadásával. (Ha a következő elemek nem látszanának, kattintsunk az *Egyebek* gombra!)
- *Kifejezéselemek listája*: Az ablak alsó felének balszélső harmadában kiválaszthatjuk azokaz az elemeket, amelyek kategóriáit majd meg szeretnénk jeleníteni. A látható kifejezéselemek eltérhetnek egymástól, attól függően, hogy a kifejezést hol és milyen célból szerkesztjük.
- *Kifejezéskategóriák listája*: Kiválaszthatjuk, hogy melyik kategóriából akarunk kategóriaelemet a Kifejezésmezőbe szúrni.
- *Kifejezésértékek listája*: Dupla kattintással a Kifejezésmezőbe visszük a kiválasztott értéket.



10. ábra: A Kifejezésszerkesztő párbeszédpanel

Az Access 2010-es változata a korábbi változatokhoz képest tartalmaz néhány új elemet. Amikor elkezdünk beírni egy függvénynevet, az *IntelliSense* legördülő listában megjeleníti a lehetséges értékeket, s dönthetünk úgy, hogy a megfelelő értékre duplán rákattintva az értéket a listából a kifejezésbe visszük be. A megfelelő elemekre kattintva az IntelliSense gyorstippeket is megjelenít, amelyek javaslatokat vagy az elemek leírását tartalmazzák.

Az IntelliSense súgót ESC billentyűvel elrejthetjük, CTRL + SPACE billentyűvel újra megjeleníthetjük.

Esetünkben nagyon egyszerű dolgunk van, hiszen – többek között – a *Now()* függvény segítségével nagyon könnyen meg tudjuk fogalmazni az érvényességi szabályt:

```
>Now()
```

A Now() függvény visszatérési értéke visszaadja az éppen aktuális rendszerdátumot és rendszeridőt. Ha ennél csak nagyobb – tehát későbbi – értéket fogadunk el, akkor az épp aktuális vagy korábbi dátum- és időértékeket kizárjuk, s nekünk épp erre van szükségünk.

A kifejezést részben vagy egészben be is gépelhetjük, de a standard eljárás a következő:

- A „>” jel beszúrása a *Kifejezéselemek: Operátorok / Kifejezéskategóriák: Összehasonlítás / Kifejezésértékek:* >út követésével történik.
- A Now() függvényt pedig a következő módon szúrhatjuk be a Kifejezésmezőbe: *Kifejezéselemek: Függvények/Beépített függvények / Kifejezéskategóriák: Dátum/Idő / Kifejezésértékek: Now*

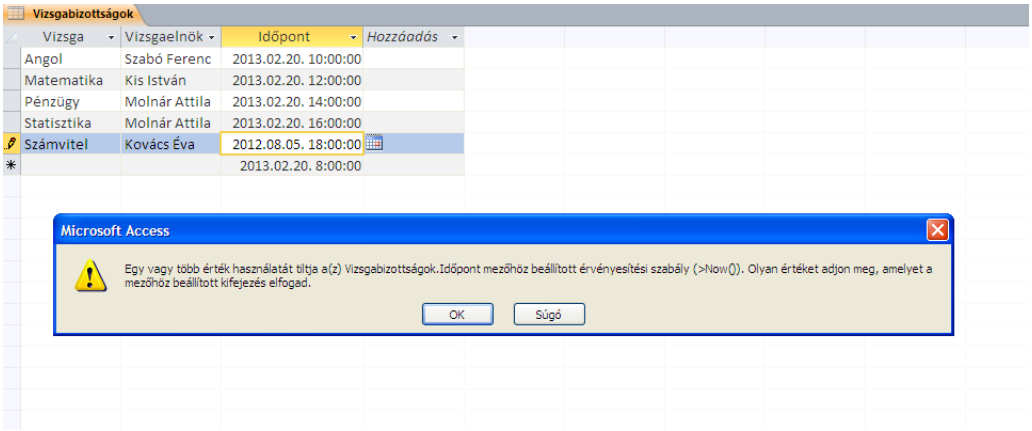
Miután a képletet bevittük a Kifejezésszerkesztőbe, kattintsunk az OK gombra!

Mielőtt az új oszlopot feltöltenénk a megfelelő értékekkel próbaképpen kattintsunk rá a menüszalagon található *Érvényességi szabályok ellenőrzése* vezérlőre (a Tervezés lap alatt)!

Miután a táblát a kapott rendszerüzenet követve mentettük, meglepve olvashatjuk az újabb rendszerüzenetet, miszerint a meglévő adatok megsértik az Érvényességi szabályt. Ez teljesen természetes, hiszen az üres, „null” értékek nem teljesíthetik azt.

Ezután mentsük el a táblát, és váltsunk Adatlap nézetre!

Vigyük be a vizsgaidőpontokhoz a következő értékeket: 2013.02.20. 10:00:00, 2013.02.20. 12:00:00, 2013.02.20. 14:00:00, 2013.02.20. 16:00:00, 2013.02.20. 18:00:00



11. ábra: Hibaüzenet az érvényességi szabály megsértése miatt

Ha a bevitt érték megsérti az érvényességi szabályt, hibaüzenetet kapunk. (11. ábra)

Az eddig tanultak alapján egészítsük ki önállóan a Vizsgabizottságok táblát egy *Maximális létszám* oszloppal, amelyhez rendeljünk Szám adattípust és Bájt mezőméretet.

Az oszlophoz tartozó adatok: 20, 25, 30, 20, 20

Lekérdezések

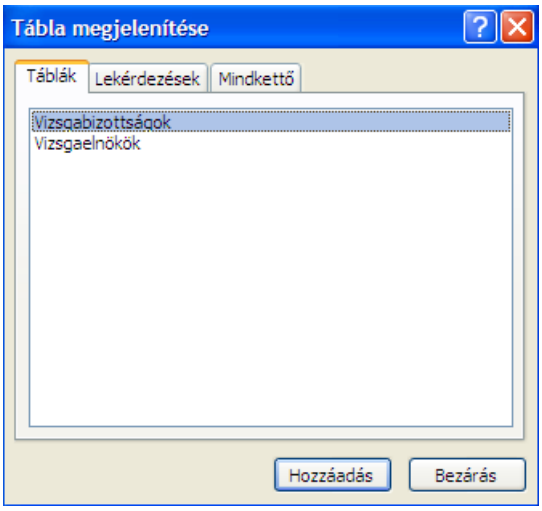
A táblákban lévő adatok listázását, szűrését, bizonyos szabályoknak megfelelő sorokban egyes adatok módosítását, törlését, más táblákba való átvitelét lekérdezésekkel tudjuk elvégezni.

A leggyakrabban használt lekérdezéstípus a *választó* lekérdezés, amelynek segítségével különböző feltételeknek megfelelő adatokat gyűjthetünk ki egy vagy több táblából.

Lekérdezés segítségével oldjuk meg a következő feladatot!

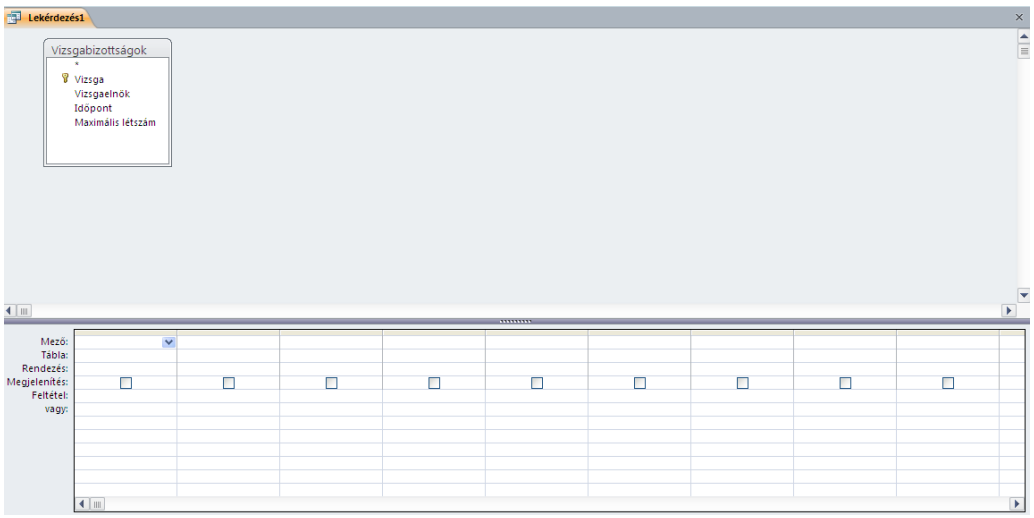
Gyűjtsük ki a táblából, hogy Molnár Attila 15 óra előtt mely bizottságokban lesz elnök!

Kattintsuk a *Létrehozás/Lekérdezéstervező* vezérlőre!



12. ábra: A Tábla megjelenítése párbeszédpanel

A felugró *Tábla megjelenítése* párbeszédpanelben válasszuk ki mindkét táblát, majd a *Hozzáadás* gombra kattintva adjuk hozzá a Vizsgabizottságok táblát a lekérdezés tervezőfelületéhez. Ezután a Tábla megjelenítése párbeszédpanelt bezárhatjuk.



13. ábra: A lekérdezés tervezőfelülete

A táblák és lekérdezések hozzáadása során a nem professzionális felhasználók tartsák be a következő két szabályt:

- Azokat a táblákat vagy lekérdezéseket adják hozzá a tervezőfelületéhez, amelyeket az adott lekérdezésben végrehajtandó művelet érint.
- Minden objektumot csak egyszer adjunk hozzá a tervezőfelülethez.

Felmerülhet a kérdés, hogy miért érdemes betartani ezeket a szabályokat.

Több praktikus ok mellett elsősorban azért, mert a hozzáadott objektumok mindegyike módosítja az SQL-kódot (az adatbázis-kezelő minden lekérdezést SQL nyelvre fordít). Néhány műveletnél ez problémát jelenthez.

A lekérdezés felületéhez tartozó táblákat a lekérdezés felső sávjában látjuk.

Ha újabb táblákat vagy lekérdezéseket akarunk megjeleníteni a tervezőfelületen, akkor vagy a *Lekérdezőeszközök/Tábla megjelenítése* vezérlőre kattintunk, vagy a felső sávba kattintunk jobb egérkapcsolóval, és a megjelenő helyi menüből választjuk ki a Tábla megjelenítése pontot.

Ha a tervezőfelülethez hozzáadott táblát vagy lekérdezést el akarjuk távolítani a felületről, kattintsunk a táblát jelképező sematikus ábrára jobb egérkapcsolóval, majd a helyi menüben kattintsunk a *Tábla eltávolítása* pontra!

A lekérdezés alapértelmezés szerint választó lekérdezés, így a *Lekérdezőeszközök/Tervezés* lapon a lekérdezés típusát nem kell módosítanunk.

A lekérdezésben megjelenített táblákat és a lekérdezéseket hordozó *Ábrarács* alatt látható *Tervezőrácson* a lekérdezés számos tulajdonsága beállítható.

A Mező paraméternél vegyük fel a tábla összes oszlopát úgy, hogy vagy kétszer az Ábrarácsban látható tábla mezőneveire kattintunk, vagy a Mező paraméter lenyíló listájából kiválasztjuk az egyes oszlopokat.

Nem kötelező minden oszlopot felvenni, de amelyik oszlopot nem vesszük fel, arra sem feltételt nem tudunk megfogalmazni, sem a hozzátartozó értékeket nem tudjuk megjeleníteni.

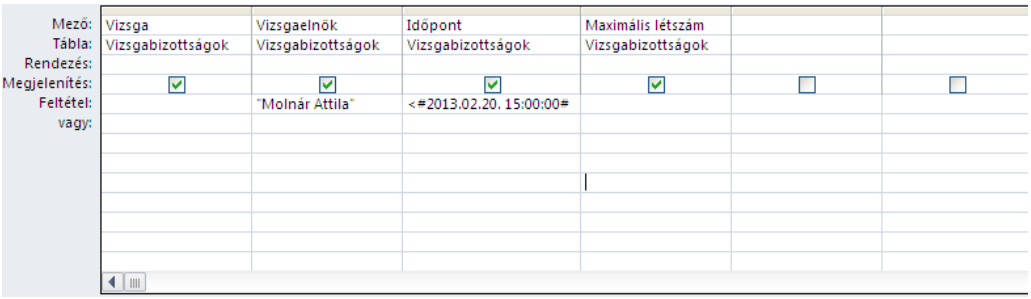
Ha valamelyik oszlopot nem akarjuk megjeleníteni, csak a rá vonatkozó feltétellel korlátozni akarjuk a megjelenítendő sorok körét, a *Megjelenítés* paraméternél látható jelölőnégyzet segítségével „eltüntethetjük”.

A Tervezőrács *Feltétel* paraméterénél adhatjuk meg a feltételeinket.

Egyszerre több oszlopra vonatkozó feltételt is megadhatunk. Ha a feltételeink a Tervezőrácson egy sorban vannak, akkor közöttük ÉS logikai kapcsolat lesz, ha nem egy sorban, akkor VAGY.

Esetünkben két, egymással ÉS logikai kapcsolatban lévő feltételünk is van.

Gépeléssel (bonyolultabb képleteknél a cellába jobb egérkapcsolóval belekattinthatunk, s a helyi menüből megjeleníthetjük a Kifejezésszerkesztőt) a Vizsgaelnök oszlop alá (a Feltétel paraméterhez) vigyük be a *"Molnár Attila"*, az Időpont oszlopba, az előző értékkel egy sorba a *<#2013.02.20. 15:00:00#* kifejezést!



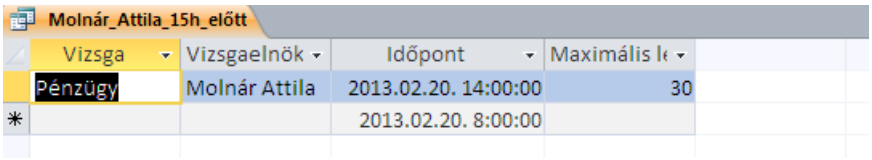
14. ábra: A Tervezőrács

A lekérdezésünk elkészült, már csak az eredményt kell megjelenítenünk.

Legegyszerűbben ezt úgy tehetjük meg, hogy a Nézet lenyíló listában az Adatlap nézetre váltunk, ezt természetesen mentés után, a lekérdezést Adatlap nézetben megjelenítve is megtehetjük.

Kattintsunk a *Lekérdezés fülre* jobb egérkapcsolóval, s a megjelenő helyi menüben a Mentés pontot választva mentsük el a lekérdezést *Molnár_Attila_15h_előtt* néven!

A lekérdezést Adatlap nézetben megjelenítve láthatjuk, hogy a lekérdezésben megfogalmazott feltételeknek összesen egy rekord felelt meg. (15. ábra)



15. ábra: A választó lekérdezés eredménye Adatlap nézetben

Az SQL és az Access

Az Access alapvetően jól használható az adatbázis-kezelésben széles körben elterjedt SQL nyelv nélkül is, azonban számos adatbázis-kezelőben szükségünk van a nyelv ismeretére. A legtöbb Access-művelet is helyettesíthető SQL-kódokkal.

Valójában minden relációsadatbázis-kezelő program – így az Access is – az SQL nyelvet használja az adatok kezelésére. Még akkor is, ha a száraz SQL-kódokat különböző felhasználóbarát eszközökkel elrejtik a felhasználó elől.

Bár rövid nyelvi bevezetőnkkel és a későbbi egyszerű példákkal ezt csak részben tudjuk megmutatni, az SQL nyelv legbonyolultabb kódjai is viszonylag könnyen érthetők még a kezdők számára is. Ha valaki ezért kedvet kap az adatbázis-kezelés mélyebb megismeréséhez, a rendelkezésre álló gazdag szakirodalomból könnyen elmélyítheti az ismereteit önállóan is.

Az SQL segítségével – egyszerű megközelítésben – adathalmazokat írhatunk le, s különböző, az adathalmazokra vonatkozó kérdéseket válaszolhatunk meg. Lényegében az eddig bemutatott és a később bemutatásra kerülő műveletek többsége leképezhető SQL-kóddal, ezt az egyes műveleteknél (elsősorban a lekérdezéseknél) könnyen ellenőrizhetjük.

Mint minden nyelvnél, az SQL nyelv alkalmazásakor is be kell tartani a szintaktikai szabályokat, tehát a helyes szintaxist kell használni. A szintaxis a nyelvi elemek helyes használatára vonatkozó szabálykészlet. Az SQL nyelv kulcsszavai az angol nyelvből származnak.

Pl. a 14. ábrán látható lekérdezés SQL-nézetben a következő:

```
SELECT Vizsgabizottságok.Vizsga, Vizsgabizottságok.Vizsgaelnök,
Vizsgabizottságok.Időpont, Vizsgabizottságok.[Maximális létszám]
FROM Vizsgabizottságok
WHERE (((Vizsgabizottságok.Vizsgaelnök)="Molnár Attila") AND
((Vizsgabizottságok.Időpont)<#2/20/2013 15:0:0#));
```

Ha a kulcsszavak jelentését lefordítjuk, s még egyszer tanulmányozzuk az eredeti lekérdezésünket, az utasítás megfejtése nem különösebben nehéz. Ismerkedjünk meg azonban röviden a SELECT utasítással, illetve néhány SQL-záradékkal!

A SELECT utasítás

Ha egy adatkészletet SQL nyelv segítségével szeretnénk leírni, akkor ahhoz SELECT utasítást használunk. A SELECT utasítás tartalmazza az adatbázisból kinyerni kívánt adathalmaz teljes leírását, tehát alapvetően a következő elemeket:

- az adatokat mely táblák tartalmazzák,
- a különböző forrásból származó adatok hogy kapcsolódnak egymáshoz,
- az adatok milyen mezők és számítások révén jönnek létre,
- feltételek, amelyek meghatározzák, hogy mely adatok kerülnek be az eredmények közé,
- szükséges-e az eredményeket rendezni, s ha igen, milyen módon.

SQL-záradékok

Az SQL-utasítások – a mondatokat kiegészítő mellékmondatokhoz hasonló módon – záradékokkal egészülnek ki. Bizonyos záradékok megadás kötelező, másoké nem. A leggyakoribb záradékok:

- **SELECT** – A visszakeresendő, kiírandó adatokat tartalmazó mezőket soroljuk fel a segítségével. Kötelező megadni.
- **FROM** – A használt mezőkhöz tartozó táblákat adjuk meg vele. Megadása kötelező.

- **WHERE** – Meghatározzuk vele azokat a feltételeket, amelyek alapján az egyes sorok bekerülnek az eredmények közé. Nem kötelező megadni.
- **ORDER BY** – Az eredmények rendezési módját állítjuk be vele. Nem kötelező.
- **GROUP BY** – Az összesítő függvényeket tartalmazó SQL utasításokban a segítségével felsoroljuk azokat a mezőket, amelyek megadják a csoportosítási szinteket. Nem kötelező megadni, viszont összesítő függvények használata esetén a SELECT-ben felsorolt, összesítő függvényben nem használt mezőket kötelező a GROUP BY záradékba besorolni.
- **HAVING** – Az összesítő függvényeket is tartalmazó SQL-utasításokban a GROUP BY záradékkal csoportosított, összesített mezőkre vonatkozó feltételeket adhatjuk meg. Hasonló a WHERE záradékhoz, csak az összesített rekordokra vonatkozik. A csoportosításból kizárni kívánt sorokat a WHERE záradékkal határozhatjuk meg, csoportosítás után pedig a HAVING záradékkal szűrhetőek a rekordok. Nem kötelező megadni.

A SELECT utasítások egymásba ágyazhatók.

Az adatbázis-tervezés második példájaként előkerülő házi könyvtár adatbázisra építve például a következő lekérdezést valósíthatnánk meg:

```
SELECT kSzerzo AS [Szerző neve], Count(kAz) AS [Könyvek darabszáma]
FROM SZERZO INNER JOIN
KONYV_SZERZO_KAPCSOLAT ON SZERZO.szKod =
KONYV_SZERZO_KAPCSOLAT.szKodWHERE kSzerzo Between "c" And "n"
GROUP BY kSzerzoHAVING Count(kAz) > 3
ORDER BY Count(kAz) DESC;
```

A lekérdezés kiválasztja a C és N kezdőbetű közé eső szerzőneveket, valamint az általuk írt könyvek darabszámát, mindezt a szerzőnevek szerint csoportosítva. Az összesítés után csak a háromnál több könyvvel rendelkező szerzőket hagyja benn, a végeredményt a könyvek csökkenő darabszáma szerint rendezi. Megfigyelhető a WHERE és HAVING záradék közötti különbség is, az első a csoportosítás előtti feltételeket tartalmazza, a második a csoportosítás utáni szűréseket végzi.

3. fejezet - Feladatok

A jegyzet egyik fő célkitűzése, hogy gyakorlatorientáltan mutassa be az adatbázis-kezelés témakörét. Ezt úgy valósítjuk meg, hogy felhasználói kézikönyv írása helyett inkább példákon keresztül mutatjuk be az adatbázis-kezelés lényegét. Ahelyett tehát, hogy azt írnánk, „ha ide kattintunk, ez történik”, „ha oda, akkor pedig az”, mi egy konkrét példa konkrét kérdéseit válaszoljuk meg, az ezekhez kapcsolódó műveletek állnak össze egésszé. Reményeink szerint ez a módszer a technikai kérdéseken túl – melyek a különböző rendszerekben eltérhetnek – a mögöttes logikát is segít megértetni. Ez utóbbi az értékállóbb ismeret, ezért a „hová kell kattintani” helyett/mellett a miért-et is próbáljuk megérteni.

A mező = tulajdonságtípus = tulajdonság = oszlop; a rekord = sor = egyedelőfordulás = egyedpéldány, valamint a tábla = egyedtípus = egyed fogalmakat szinonimaként kezeljük.

Adatbázisok tervezése – példák

A legtöbb szakkönyv ezt a kérdést jótékonyan átugorja és mindjárt olyan példákkal indít, amelyekben a különböző táblák és tulajdonságok meg vannak határozva. A gyakorlatban ez nem így van, ezért mi fontosnak tartjuk néhány példa bemutatását a tervezés kapcsán is.

1. példa

Tanfolyam nyilvántartó adatbázist építünk. Meghatároztuk a tárolásra kerülő adatok körét, ezek az alábbiak:

Rövid kód	Leírás
tAzon	A tanfolyam azonosítószáma
hCim	A hallgató címe
tKDat	A tanfolyam kezdő dátuma
hTel	A hallgató telefonszáma
tDij	A tanfolyam díja
hNev	A hallgató neve
tBefj	Befejezettség jelző (befejeződött-e a tanfolyam?)
hAzon	Hallgató azonosítószáma
fizJ	Fizetés jelző (fizetett-e a hallgató)
tMegn	A tanfolyam megnevezése
fizMod	Fizetés módja (miként fizet a hallgató?)
hSzdDat	A hallgató születési dátuma

15. táblázat: 1. példa – kiinduló tábla

Határozzuk meg az egyedtípusokat, állítsuk fel az adatmodellt!

(Az áttekinthetőség kedvéért most rövid kódokkal dolgozunk, melyek a leírás szavaiból álltak össze. Az első szó kivételével a szókezdő betűk nagy betűk, segítve a kódok értelmezését.)

Megoldás:

Az adatok körének meghatározása során kirajzolódik, hogy milyen önálló objektumokra számíthatunk a rendszerben. Jelen esetben első lépésben két egyedtípusra oszthatjuk az adathalmazt: hallgató, tanfolyam. Különítsük el így az adatokat:

HALLGATO	
hAzon	Hallgató azonosítószáma
hNev	A hallgató neve
hSzdDat	A hallgató születési dátuma
hCim	A hallgató címe
hTel	A hallgató telefonszáma
TANFOLYAM_TORZS	
tAzon	A tanfolyam azonosítószáma
tMegn	A tanfolyam megnevezése
tKDat	A tanfolyam kezdő dátuma
tDij	A tanfolyam díja
tBefj	Befejezettségjelző (befejeződött-e a tanfolyam?)
fizJ	Fizetésjelző (fizetett-e a hallgató)
fizMod	Fizetés módja (miként fizet a hallgató?)

16. táblázat: 1. példa – segédtábla

Felmerül a kérdés, hogy hová kerüljön a fizetésjelző és a fizetés módja mező (tulajdonság), a hallgatóhoz vagy a tanfolyamhoz? Első ránézésre akár mindkettőhöz mehetne, hiszen a hallgató fizet és tanfolyamot fizet ki. Hol a probléma? Kössük össze képzeletben a két adatbázist. Milyen kapcsolat van közöttük? Egy hallgató akár több tanfolyamon is részt vehet, egy tanfolyamon pedig nyilván több hallgató tanul. Ez több-a-többhöz (M:N) típusú kapcsolat, azaz egy *illesztőtáblára* lesz szükségünk.

Van egy HALLGATO táblánk, és egy törzsadatokat tartalmazó TANFOLYAM_TORZS táblánk. Utóbbi a tanfolyam alapadatait tartalmazza, mely független a hallgatóktól (nem úgy például a fizetés módja, mely az egyik hallgatónál pl. csekkes, a másiknál átutalás, stb.):

```
HALLGATO (hAzon, hNev, hSzdDat, hCim, hTel)

TANFOLYAM_TORZS (tAzon, tMegn, tKDat, tDij, tBefj)
```

E mellé kerül az illesztőtábla, mely a konkrétan megadja a hallgatók és tanfolyamok kapcsolatát:

```
TANFOLYAM (hAzon, tAzon, …)
```

E táblát a hallgató és a tanfolyam azonosítója együttesen határozza meg, a két azonosító együtt alkot összetett elsődleges kulcsot. A két mező egyértelműen azonosítja, s egyben meg is különbözteti egymástól az *egyedpéldányokat* (egyed-előfordulásokat, vagy sorokat). A fizetés jelző és fizetési mód adott hallgató és adott tanfolyam esetén egyedi, így e két elem a TANFOLYAM egyedbe illeszkedik. A végeredmény tehát:

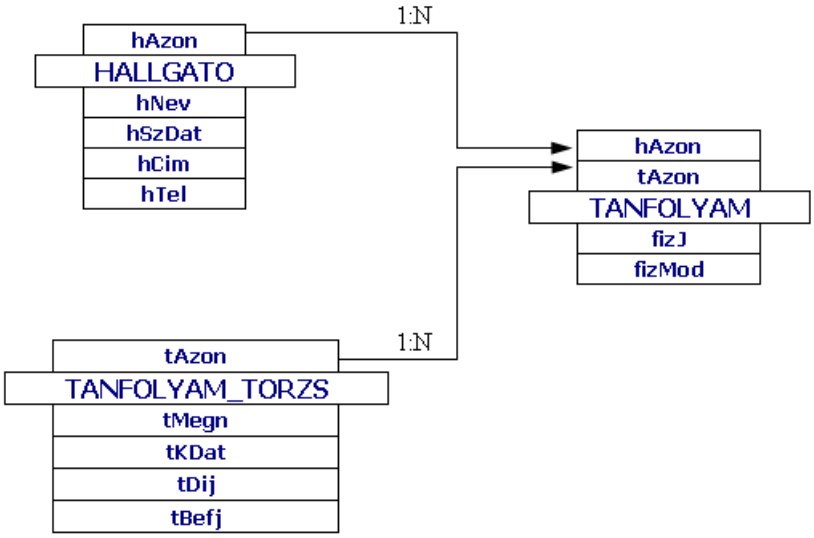
```
HALLGATO (hAzon, hNev, hSzdDat, hCim, hTel)

TANFOLYAM_TORZS (tAzon, tMegn, tKDat, tDij, tBefj)

TANFOLYAM (hAzon*, tAzon*, fizJ, fizMod)
```

Ha végigmegyünk a normalizálási szabályokon, láthatjuk, hogy nincsenek *ismétlődő csoportjaink* (1NF). Minden nem elsődleges kulcsként funkcionáló tulajdonság teljes funkcionális függőségben van az elsődleges kulcs egészétől (2NF – ez a TANFOLYAM táblánál érdekes, hiszen csak ott van összetett kulcsunk: mind a fizJ, mind a fizMod az elsődleges kulcs egészétől függ). Továbbá a nem elsődleges kulcsként funkcionáló tulajdonságok között nincs funkcionális függés (3NF). Az adatbázis tervezését lezárhatjuk.

Az előzőekkel ellentétben itt egy ábrát is elhelyezünk a biztosabb megértés érdekében. Az egyedek nevei nagyobb téglalapba kerültek. Felettük az elsőleges kulcsot alkotó tulajdonságok, alattuk a nem elsődleges kulcs mezők. A kapcsolatot nyilak ábrázolják, a kapcsolat típusát a nyilakra írtuk. A nyilak a „több” irányba mutatnak.



16. ábra: 1. példa – a végeredmény grafikusan

2. példa

Egy házi könyvtár adatbázisát építjük fel. Meghatároztuk, hogy az egyes könyvekről milyen adatokat szeretnénk eltárolni:

Rövid kód	Leírás
isbn	ISBN szám, egyedi könyvazonosító
koCim	A kiadó címe
kiCim	A könyv címe
kiNev	A kiadó neve
kEv	A könyv kiadásának éve
kTemakor	A könyv témaköre, műfaja
kBeDatum	A könyv beszerzésének dátuma
kDij	Beszerzési ár (Ft)
kSzerzo	A szerzők(k) neve(i)
kTerj	A könyv oldalainak száma

17. táblázat: 2. példa – kiinduló tábla

Határozzuk meg az egyedtípusokat, írjuk fel az adatmodellt!

Megoldás:

Első ránézésre akár az egész tulajdonsághalmazt egy táblába szervezhetnénk. Ha elegendő egy táblázat, akkor táblázatkezelésről beszélünk, nem adatbázis-kezelésről. Nincsenek kapcsolatok, az egyetlen tábla karbantartását, lekérdezését pedig táblázatkezelő szoftverben (pl. Microsoft Office Excel) tudjuk a legkényelmesebben elvégezni. Bár az Excelben is számos művelet adatbázis-műveletként szerepel,

ezek mindegyike az adott táblán belül hat, így az ottani elnevezés csalóka. Itt azonban – ha alaposabban tanulmányozzuk az adatokat, kiderül, hogy – nem lesz elegendő egy tábla.

A házi könyvtárban feltételezhetően lesz egy KONYV egyed. Nézzük, mely tulajdonságok tartozhatnak ide:

KONYV	
isbn	ISBN szám, egyedi könyvazonosító
kiCim	A könyv címe
kEv	A könyv kiadásának éve
kTerj	A könyv oldalainak száma
kBeDatum	A könyv beszerzésének dátuma
kDij	Beszerzési ár (Ft)
kTemakor	A könyv témaköre, műfaja
kSzerzo	A szerzők(k) neve(i)

18. táblázat: 2. példa – segédtábla

Ide tartozik a könyv azonosítójaként használt ISBN szám, a könyv címe, kiadásának éve, oldalainak száma. Ha egy konkrét könyvet leveszünk a polcról és felütjük, ezek pontosan kiolvashatóak belőle, tehát egyértelműen hozzá tartozó adatok. Az adott könyv beszerzésének dátuma és beszerzési ára szintén egy konkrét könyvhöz köthető adat. A témakör a könyv műfaját, pl. krimi, sci-fi, stb. jelöli meg, egyfajta tematikus címke. A szerző is szerepel a felütött konkrét könyvben, csak épp egy könyvnek több szerzője is lehet.

Ha a több szerzőt egyetlen tulajdonságértékként vesszük fel, akkor Horváth Balázs könyveinek lekérésekor nem jön elő Kiss Péterrel közösen írt könyve, hiszen ott „Horváth Balázs” helyett a „Horváth Balázs – Kiss Péter” érték szerepel. Ha ez számunkra fontos, akkor a szerzőket külön táblába kell szervezni. Egy szerző több könyvet is írhatott, illetve egy könyv több szerzővel is rendelkezhet, tehát több-a-többhöz kapcsolatról beszélünk, szükségünk lesz egy illesztőtáblára. Ezen felül a szerzőnév önmagában nem szerencsés elsődleges kulcsnak. Az írók törekednek az önállóságra így kevés egyforma nevű szerző van – kivételt képeznek pl. Alexandre Dumas-ék, akik viszont idősebb és ifjabb jelzővel választották el a nevüket. Itt mindenképpen szükség van elhatárolásra, hogy a könyvek adatainak feltöltésénél egyértelmű legyen, melyik szerzőt kell felvinni, különben különböző életműveket fogunk egybeszerkeszteni. Szóval azt feltételezhetjük, hogy a szerzőnév-adatok egyediek (és azt is, hogy az adatok feltöltője tudni fogja, hogy az egyes könyveket mely szerző(k)höz rendelje hozzá). Viszont a kulcsként használt feleslegesen hosszú adatok növelik a tárigényt és lassítják a feldolgozást, ezért vezessünk be szKod néven a szerző kódja tulajdonságot:

```
KONYV (isbn, kiCim, kEv, kTerj, kBeDatum, kDij, kTemakor)

SZERZO (szKod, kSzerzo)

KONYV_SZERZO_KAPCSOLAT (isbn*, szKod*)
```

Tehát a könyveket csak egyszer rögzítjük, a szerzőket is csak egyszer, és a köztük lévő kapcsolatokat azonosítóik kombinációjával határozzuk meg.

Nézzünk rá egy pillanatra a kTemakor tulajdonságra. Ha a műfaji címkét lazábban értelmezzünk, akkor egy könyv akár több címkét is kaphat. Lehet pl. krimi és regény, vagy novella, de ha címkézni szeretnénk, akár még a vallás, gasztronómia, néprajz, stb. szavak is beleférhetnek e tulajdonságba. Talán szerencsésebb lett volna egyből kulcsszónak, vagy címkének nevezni. Ha így vesszük, e kategorizálásra alkalmas mező kapcsán is több-a-többhöz kapcsolatról beszélhetünk. A megoldás ugyanaz, mint a szerző neve mezőnél. Önálló egyed felvétele TEMAKOR néven, illetve egy új azonosító kTKod néven, valamint a kapcsolatot megvalósító illesztőtábla létrehozása:

```
KONYV (isbn, kiCim, kEv, kTerj, kBeDatum, kDij)

SZERZO (szKod, kSzerzo)

KONYV_SZERZO_KAPCSOLAT (isbn*, szKod*)

TEMAKOR (kTKod, kTemakor)

KONY_TEMAKOR_KAPCSOLAT (isbn*, kTKod*)
```

Ha csak egy témakör lenne hozzárendelhető az egyes könyvekhez, de igen sok könyvünk van, viszont kevés, ám hosszú nevű témakörünk, akkor is érdemes lenne elgondolkodni az önálló egyedbe szervezésen, hiszen a létrejövő egy-a-többhöz kapcsolat révén egy rövid kóddal kötjük a témaköröket az egyes könyvekhez, ami adattárolási szempontból előnyös.

A kiadó adatait már nem is szerepeltettük a KONYV egyed lehetséges mezőit tartalmazó segédtáblában. Egy-egy kiadó több könyvet is kiadhatott, egy könyv pontosan egy kiadóhoz köthető, ez tehát egy-a-többhöz kapcsolat. Másként gondolkodva: ha a kiadó neve (kiNev) és kiadó címe (kiCim) tulajdonságokat besoroltuk volna a KONYV tulajdonságai közé, a kiCim nem az elsődleges kulcstól, hanem a kiNev tulajdonságtól függne, tehát nem teljesülne a harmadik normálforma (3NF).

A megoldás, egy önálló KIADO egyedtípus létrehozása. Ehhez a korábban felvázolt okok miatt felveszünk kiKod néven egy kiadó kódja mezőt is:

```
KIADO (kiKod, kiNev, kiCim)
```

A KIADO áll az „egy” oldalon, több KONYV egyedelőforduláshoz lehet ugyanazt a kiadót hozzárendelni, így a kiadó kódját tesszük hozzá a KONYV egyedtípushoz. Azért, hogy egyértelmű legyen, nem a megnevezéseknek, hanem a tartalmaknak kell megegyezniük, a KONYV egyedtípusba nem kiKod, hanem a könyv kiadójának azonosítója, kKAz néven vesszük fel a kérdéses kódot, de tartalmilag az itt szereplő kódok a KIADÓ tábla kiKod mezőjében lévő kódoknak felelnek meg, a kKAz lesz a kapcsolat idegen kulcsa:

```
KONYV (isbn, kiCim, kEv, kTerj, kBeDatum, kDij, kKAz*)

SZERZO (szKod, kSzerzo)

KONYV_SZERZO_KAPCSOLAT (isbn*, szKod*)

TEMAKOR (kTKod, kTemakor)

KONY_TEMAKOR_KAPCSOLAT (isbn*, kTKod*)

KIADO (kiKod, kiNev, kiCim)
```

Ezzel el is készültünk. Vagy mégsem?

Az ISBN-kód minden könyvön egyedi, azonban a könyvek egyes példányaira ez nem igaz. Ha valamiből több példányunk van és mindegyiket rögzítenénk az adatbázisban, akkor a fenti adatszervezés nem lesz megfelelő, hiszen a KONYV tábla két ugyanolyan isbn – elsődleges kulcs – kerülne. Ez pedig nem lehetséges. Mi lehet a megoldás?

Az egyik, hogy a KONYV táblához felveszünk egy darabszám tulajdonságot is. Ezzel az azonos című könyveknél jelezhetjük, hogy több van belőlük. A könyvtárakban is hasonló rendszer van. Igen ám, de ha a cím, a kiadó, vagy pl. a kiadás éve meg is egyezik az azonos könyveknél, a beszerzés dátuma, illetve költsége lehet eltérő az egyes eseteknél, és ezt ez a megoldás nem kezeli megfelelően. Könyvtáraknál a kérdéses adatok nem szerepelnek a katalógusban, ott ez a probléma nem lép fel.

A másik megoldás, hogy a KONYV táblához felveszünk egy egyedi könyv azonosítót (kAz), ezt akár a könyvespolcon lévő könyvek gerincére is felragaszthatjuk, így az elektronikus adatbázis és a polcon

lévő könyvek között olyan kapcsolatot építhetünk, amely logikus kódolás és ennek megfelelően rendezett könyvállomány esetén, a polcon történő kikeresést is segíti. Nincs szükség darabszámra, az azonos ISBN-kódok alapján ez kiszámítható. Ez esetben értelemszerűen a kapcsolódó idegen kulcsokat is illeszteni kell. Joggal feltételezhetjük, hogy a legtöbb könyvből egy példány lesz a házi könyvtárban, és egy-egy könyv több példánya esetén sem beszélhetünk komoly nagyságrendről, maximum pár darabról, így további alakításra nincsen szükség.

KONYV (**kAz**, isbn, kiCim, kEv, kTerj, kBeDatum, kDij, kKAz*)

SZERZO (**szKod**, kSzerzo)

KONYV_SZERZO_KAPCSOLAT (**kAz *, szKod***)

TEMAKOR (**kTKod**, kTemakor)

KONY_TEMAKOR_KAPCSOLAT (**kAz** *, **kTKod***)

KIADO (**kiKod**, kiNev, kiCim)

3. példa

Egy folyószámla könyvelési rendszerben az alábbi információkat szeretnénk kezelni. (Természetesen egy banki rendszer jóval bonyolultabb, nekünk viszont elegendő most a megadott adatok lehetőségein belül gondolkodnunk.)

Rövid kód	Leírás
uKod	Ügyfél azonosítószáma
uNev	Ügyfél neve
uAdosz	Ügyfél adószáma
uIrsz	Ügyfél címének irányítószáma
uTelep	Ügyfél címének településrésze
uUtca	Ügyfél címének utcarésze
uTel	Ügyfél telefonszáma
fszla	Számlaszám
fAktiv	Aktív-e a számla
fNyitD	Folyószámla megnyitásának dátuma
fEgyenl	Folyószámla egyenlege
tKod	Tranzakció azonosító kódja (kifizetés, utalás, befizetés, bejövő utalás… - másként a tranzakció fajtája)
tOsszeg	Tranzakció összege
tDatum	A tranzakció dátuma
ellTul	Az ellenszámla tulajdonosának neve (utalás esetén)
ellSzla	Az ellenszámla száma
kBizSzam	Könyvelési tétel bizonylatszáma
tKozl	A tranzakció közleménye
kRendben	Könyvelési jelző (rendben van vagy nincs rendben a tranzakció)

19. táblázat: 3. példa – kiinduló tábla

Határozzuk meg az egyedtípusokat, írjuk fel az adatmodellt!

Itt nem bocsátkozunk hosszú fejtegetésbe, ezt a feladatot önálló fejtörésre szánjuk. Álljunk meg az olvasással, próbáljuk megoldani és csak utána olvassunk tovább a szöveget!

Ez egy összetettebb példa, ezért nem kevertük össze a tulajdonságtípusokat, az egyes egyedek tulajdonságai egymás alatt sorakoznak.

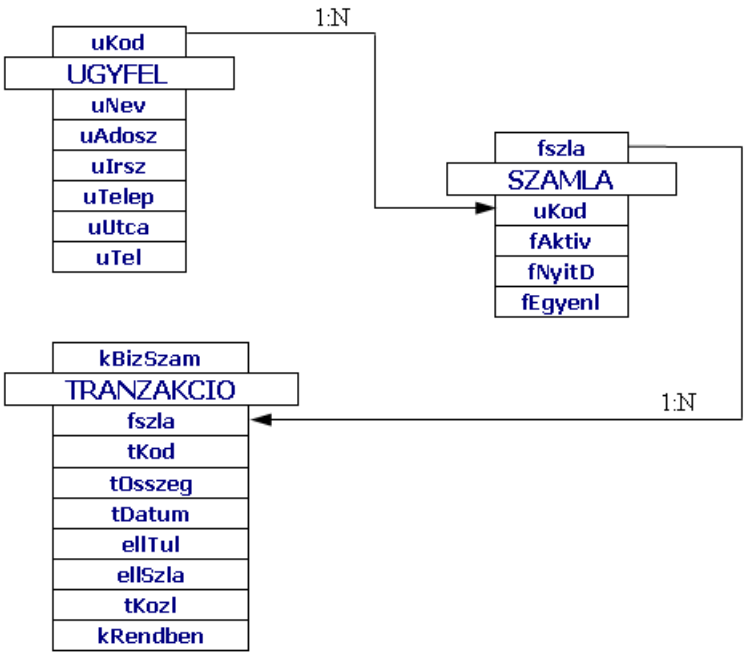
Ha megfigyeljük, a felső rész az ügyfél adatait takarja, így célszerű egy ügyfél egyedet létrehozni. Azonosítóként adja magát az uKod – ügyfél azonosítószáma, ami egy bankon belüli azonosító jel. Választhattuk volna az adószámot is, viszont akkor mindenkinek kellene adószámot megadnia, ami adott esetben nehézkes lehet (pl. külföldieknél). Egyszerűbb tehát egyedi generált azonosítót használni.

Az ügyfeleknek vannak számláik, amelyeken tranzakciókat bonyolítanak le. Egy ügyfél több számlával is rendelkezhet (egy-a-többhöz kapcsolat), egy számlán pedig több tranzakciót is végre lehet hajtani. A másik két egyed a számla (amely a számlához tartozó, egyszer megjelenő tulajdonságtípusokat tartalmazza, pl. nyitás dátuma) és a tranzakció. Hosszabb kifejtés nélkül a következő ábrán közlünk egy lehetséges megoldást:

```
UGYFEL (uKod, uNev, uAdosz, uIrsz, uTelep, uUtca, uTel)

SZAMLA (fszla, uKod*, fAktiv, fNyitD, fEgyenl)

TRANZAKCIO (kBizSzam, fszla*, tKod, tOsszeg, tDatum,
ellTul, ellSzla, tKozl, kRendben)
```



17. ábra: 3. példa – a végeredmény grafikusan

Adatbázis-kezelés – példa

A példa szövege

A Fangorn Sörgyár a HORECA-területen (Hotel – Restaurant – Cafe; szálloda- és vendéglátóipar) történő értékesítéshez HORECA-képviselőket alkalmaz. A képviselők a szerződések alapján kapnak jutalékot, a jutalék százalékos mértéke a képviselőkkel történő egyéni megállapodásoktól függ. A

HORECA-vállalkozások számára a cég háromféle szerződési opciót kínál. Ha vállalják, hogy kizárólagosan a sörgyár termékeit árusítják, akkor kedvezményre jogosultak. Ezen felül választhatnak három opció közül, melyek 2% extra engedményt, vagy meghatározott értékhatárig az esetlegesen túlkészletezett és lejárt termékek visszavásárlását, vagy egy rendezvény termékalapú szponzorálását jelentik egy adott évben. Ezek kódjai OP01, OP02, illetve OP03 lesznek. A HORECA-szerződésekben szerepel egy éves minimális vállalási összeg is, melyet a szerződő az adott évre vállal. Ettől is függ az alapból járó kedvezmény mértéke és a képviselő is ezen összeg alapján kap jutalékot.

Az adatokat számítógépes rendszerben kezeljük. Meghatározásra kerültek a rögzítendő adatok és egyedekbe szervezésük is megtörtént. Ezen felül az alábbi táblázatban az adatokkal kapcsolatos különféle megkötéseket is láthatjuk.

A KEPVISELO egyedtípus tulajdonságai:		
Tulajdonságtípusok		**Megkötések**
Név	**Szöveges értelmezés**	
KAZON	A képviselő azonosítója	Pontosan 5 számjegy karaktert tartalmazó egyedi azonosító
KNEV	Név	Kötelező adat
KTEL	Telefonszám	-
KJUT	A jutalék mértéke ezrelékben	0-nál nagyobb, 50-nél kisebb
A HORECA egyedtípus tulajdonságai:		
Tulajdonságtípusok		**Megkötések**
Név	**Szöveges értelmezés**	
HAZ	Szerződésazonosító	A megkötött szerződés egyedi azonosítója, a rendszer által generált
HNEV	A cég neve	Kötelező adat
HCIM	A cég címe	-
HCEGDAT	A cégbejegyzés dátuma	A cég minimum kétéves kell, hogy legyen, az adat megadása kötelező.
HOP	Szerződés extra opciója	Lehetséges értékek: OP01, OP02, OP03
HKOTDAT	A szerződés megkötésének dátuma	Alapértelmezésben a rendszerdátum
HKEZDAT	A szerződés életbe lépésének dátuma	Nagyobb-egyenlő a szerződéskötés dátumánál
HMINOSSZ	A vállalt éves minimális rendelési összeg	1.000.000-nél nagyobb, 100.000.000-nál kisebb A megadott összeg legyen osztható százezerrel és Ft-ként jelenjen meg (ez csak megjelenésre vonatkozik!)
KAZON	A szerződést megkötő képviselő azonosítója	A KEPVISELO egyedtípusból

20. táblázat: Adatbázis-kezelés példa – kiinduló táblák

Itt az azonosítók nyomtatott nagybetűvel kerültek rögzítésre, hogy a számítógépes megvalósításkor, illetve itt a szövegben könnyen felismerhetőek legyenek.

Adatbázis-kezelőként továbbra is a Microsoft Access szoftvert használjuk. A szoftver alapjainak korábbi felvázolása után most egy hosszabb példán keresztül mutatjuk be az adatbázis-kezelés lehetőségeit. Itt újra az alapoktól indulunk, amit már az „alapozó” részben említettünk, arra csak visszautalunk.

Az elkészült adatbázis állományt ide kattintva töltheti le. [images/HORECA_nyilvantartas.accdb]

Ha még nem jutott végig a feladatokon, mindenképpen nézze meg a 20. feladat beállításait. Ezek „inverzével” tudja az adatbázis meg nem jelenő objektumait visszahelyezni a navigációs panelre.

1. feladat – az adatbázis létrehozása

Ellenőrizze a fenti struktúrát, majd hozza létre a HORECA_NYILVANTARTAS adatbázist!

Első lépésben gondoljuk át a fenti struktúrát. A KEPVISELO egyedtípusba jóval több adat is kerülhetne – pl. beosztás, e-mail cím, stb. – de a feladat szövege szerint csak a fenti adatok szükségesek. (Nyilván azért, hogy a feladat belátható méretű maradjon.) Elsődleges kulcsként a KAZON tulajdonságtípust érdemes választani, a többi mező ettől, és csak ettől függ, ismétlődő csoportok nincsenek (feltételezzük, hogy minden képviselő egy céges mobilszámmal rendelkezik, legalábbis nekünk ez az egy adat elég, többet nem szeretnénk rögzíteni). Itt tehát teljesül a harmadik normálforma is.

A HORECA egyedtípus elsődleges kulcsa a szerződés azonosítója. A szerződéshez kötődő adatok (pl. szerződés megkötésének dátuma, vagy a vállalt éves minimális rendelési összeg) értelemszerűen függésben vannak az azonosítótól. Kérdés lehet a cégadatok önálló egyedbe szervezése. Tegyük fel, hogy egy cég nem köthet több HORECA-szerződést. Egy szerződéshez egy cég tartozik, és fordítva, egy céghez egy szerződés rendelhető, a kapcsolat tehát egy-az-egyhez, nincs ismétlődő csoport. Ha integrált sörgyári rendszerben dolgozunk, akkor persze a VEVO törzsadatcsoport számos helyen használható lenne a rendszerben, így érdemes lenne külön egyedként megjeleníteni és a HORECA egyedhez is idegen kulccsal kapcsolni. Ettől most eltekintünk, feltételezzük, hogy a feladatban felvázolt adatbázis semmilyen külső kapcsolattal nem rendelkezik és ilyet nem is terveznek, továbbá egy cég csak egy szerződést köthet. Még ebben az esetben is érdemes lehet egy külön SZERZODO egyedet létrehozni a HNEV, HCIM, HCEGDAT tulajdonságtípusokkal, mivel az adott cég új, és új szerződésben újíthatja meg szerződését. Belátva a fentieket, mi most ennek ellenére maradunk a feladatban felvázolt két egyedtípusnál.

Egy KEPVISELO több HORECA-szerződést is köthet, ezért a két tábla között egy-a-többhöz kapcsolat van.

Ezután létrehozhatjuk az adatbázist, ennek menetéről már a bevezető feladatsorban esett szó.

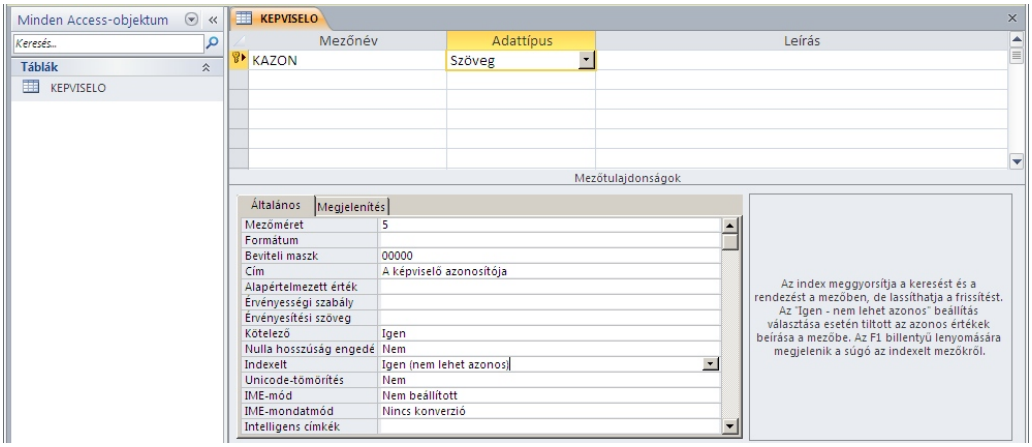
2. feladat – az adatbázistáblák létrehozása

Tervezze meg, majd hozza létre az adatbázis adattábláit a megkötéseknek megfelelően (kezdőérték, felirat, index, formátum, ellenőrzés, stb.)! Hozza létre a táblák közötti kapcsolatot!

Vegyük fel a KEPVISELO táblát – az automatikusan megjelenő Tábla1 átnevezésével – a korábban megismert módon! Tervező nézetben vegyük fel az első mezőt (KAZON)! A mezőnévhez a rövid kódot írjuk, ezt csak mi fogjuk látni. A kifelé mutatott megnevezést – jelen esetben: a képviselő azonosítója – a mező cím tulajdonságába írjuk bele! Mi legyen a mező adattípusa? Az adatok öt számjegyből állnak. Az egész altípus két bájton tárolja a számokat, felső értéke 65535, azaz nem férne bele az összes öt számjegyű szám, marad tehát a négy bájtot foglaló hosszú egész altípus.

Ezzel azonban több gondunk is van. Egyfelől nem kezeli megfelelően a nullával kezdődő kódokat, a kezdő nullákat egyszerűen törli, bár a bevitelt engedi. Másfelől a kódok összehasonlítását bonyolítja, hogy a számok nem karakterenként tárolódnak. Pl. ha az 5-tel kezdődő kódokat szeretnénk lelistázni, akkor nem mondhatjuk szűrőfeltételnek, hogy 5* – a „csillag” meghatározatlan számú és tetszőleges karaktert jelent – hanem a szám nagyságrendjéből kell kiindulnunk, és a >=50000 and <60000 feltétellel kellene élnünk. Ha a kódokban valamilyen logika van – mint például az irányítószámokban – a számjegyek összehasonlítását a számként tárolás feleslegesen megnehezíti. Mit nyernénk cserébe? A számok összeadhatóak, a szövegek nem. Általánosítva: a kódokkal matematikai műveleteket tudnánk végezni. Erre itt nincsen szükség. Célszerű tehát a szöveg adattípust választani, mellyel karakterenként tárolódik a megadott öt számjegy. A típus meghatározásánál a felhasználható karakterek típusa helyett mindig a tartalmi hasznosságot vizsgáljuk. Ökölszabályként elmondható, hogy ha a számjellegű

értékekkel nem szeretnénk matematikai műveleteket végezni – nincs értelme az összegüknek –akkor tároljuk őket szövegként (lásd pl. az irányítószámot is).



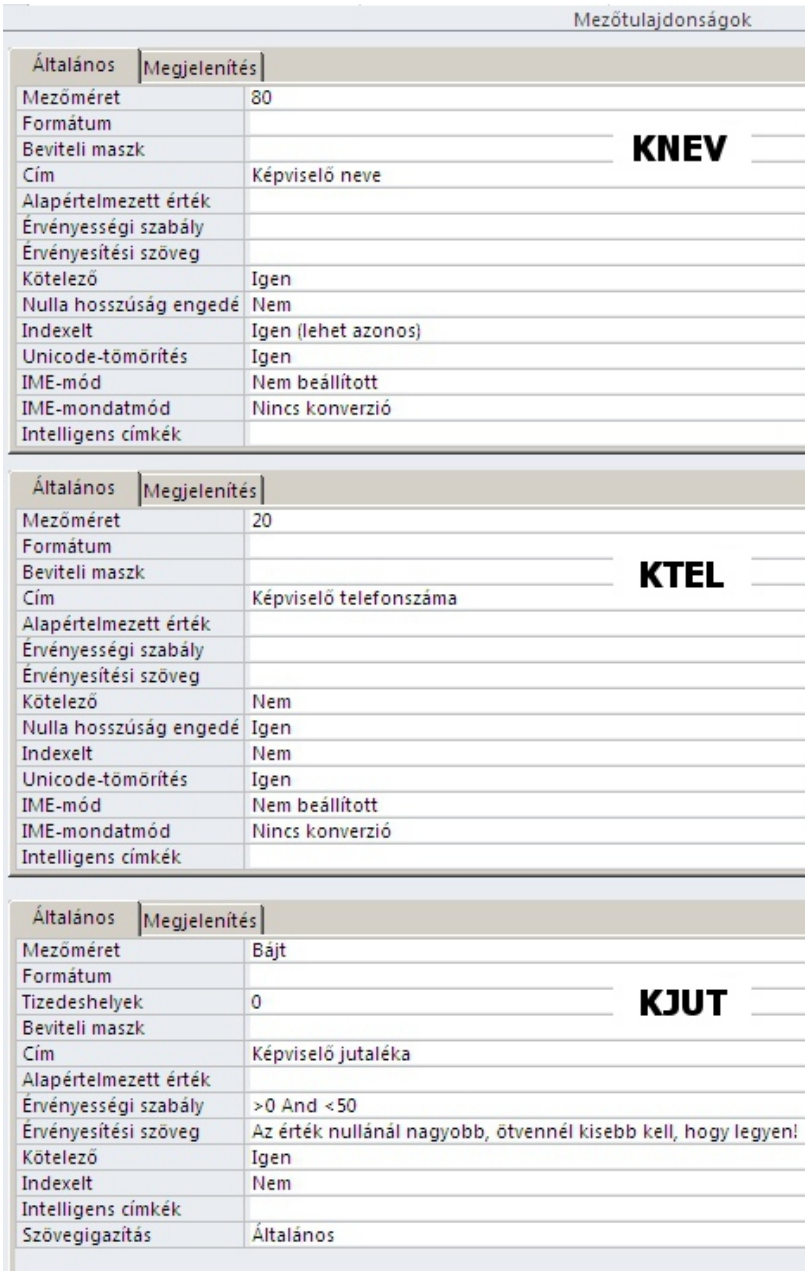
18. ábra: 2. feladat – a KAZON tulajdonságtípus mezőtulajdonságainak beállításai

A feladatban az a kitétel is szerepel, hogy „tervezze meg”. A példa szövegében néhány megkötéssel éltünk, de nem volt cél a tulajdonságtípusok különböző mezőtulajdonságainak teljes körű meghatározása. Ezt jelen feladat „tervezze meg” kitételével az olvasóra bíztuk. Itt van például az indexelés kérdése. Az indexeléssel a számítógép tulajdonképpen egy folyamatosan karbantartott listát rendel az adott mező mellé, mely az adott mezőben szereplő értékek sorrendjét adja meg. Ez azt jelenti, hogy minden újabb rekord felvitelekor minden indexelt mező esetén a számítógépnek az új adatot be kell illesztenie az indexlistába, ami lassíthatja a frissítést. Cserébe azonban jelentősen gyorsulhat a keresés és a rendezés az adott mezőben. Azt kell tehát eldöntenünk, hogy az adott mező felől szeretnénk-e keresést indítani, illetve igen válasz esetén kérdés még, hogy felvehet e több rekord is azonos mezőértéket („Igen, lehet azonos”, illetve „igen, nem lehet azonos” mezőtulajdonság értékek). A táblák kapcsolatait megvalósító tulajdonságtípusokat mindig érdemes indexelni, mivel ezeken keresztül biztosan történik keresés, ha egy másik tábla felől indulva szeretnénk a kérdéses táblából adatokat megtudni. Az indexek segítségével történő keresések és lekérdezések hatékonysága annál nagyobb, minél több különböző mezőérték található egy adott táblában. Kevés különböző érték nagyszámú előfordulásakor az indexelés kevésbé hatékony. Például, ha egy könyvben tárgymutató található (ez az index), az nagy segítség pl., hogy megtudjuk, hol használták az integritás szót. Megnézzük a tárgymutató I betűjénél a kérdéses szót, és az oldalszámokat, ahol előkerül, majd odalapozunk és már látjuk is, milyen szövegkörnyezetben, hogyan került elő. Mi a helyzet, ha nem az integritás szót, hanem az a és az névelőket keressük? Hiába az alapos tárgymutató, ha ezek kigyűjtésre kerültek, akkor sem lesz túl nagy segítségünkre a vélhetően minden oldalszámot felsoroló lista.

Legyünk most például kíváncsiak arra, hogy egy adott képviselő – akinek a nevét tudjuk – milyen szerződéseket kötött. Ilyen kérdés gyakran előfordulhat, a gyors előkereséshez érdemes lesz tehát a KNEV mezőt indexelni. Lehetnek azonos nevű képviselőink, tehát „igen (lehet azonos)” lesz a helyes beállítás. Egyforma nevek és megfelelően megszerkesztett lekérdezés esetén a rendszer ezt jelezni fogja. Érdemes a HORECA táblában lévő KAZON tulajdonságtípust is indexelni, mivel ezen keresztül jutunk át az egyes szerződésekhez. Figyeljünk, hogy itt is „igen (lehet azonos)” lesz a helyes beállítás, hiszen idegen kulcsként több szerződés is kapcsolódhat egy képviselőhöz. Kíváncsiak lehetnénk arra is, hogy egy adott városban milyen szerződéseink vannak. Ehhez a HCIM – szerződő partner címe – tulajdonságtípusban például a Sopron településnévre kellene szűrnünk. Erre jelen felépítésében az adatbázis csak korlátozottan alkalmas, mivel a teljes cím egy mezőbe kerül és semmilyen kötelező formai előírást sem tudunk alkalmazni – például, hogy irányítószám, település, utca, házszám struktúrában kerüljenek bevitelre a címek, a szóközök, vesszők, pontok kérdéséről már nem is beszélve. Ha ilyen kérdésekre is választ szeretnénk, érdemesebb lett volna a cím egyes részeit – irányítószám, település, utca és házszám – külön tulajdonságtípusokba szervezni. Jelen pillanatban a *Sopron* szűréssel ki lehet listázni a soproni szerződéseket, de közéjük kerül például a Soprontól távoli Kétsoprony is. A listázást pedig nem segíti az indexelés beállítása, mivel a szöveges adattípust ábécé szerint rendezi, viszont nem biztosított, hogy a cím felvitele a településnévvel, vagy az irányítószámmal kezdődjön, így a különböző soproni címek az indexelés ellenére is távol kerülhetnek egymástól.

Feltételezzük azt, hogy a példa adatbázisát azért szervezték így, mert az eddig fejtegetett lekérdezésekre nincsen szükség.

A KEPVISELO tábla következő tulajdonságait rendre az alábbi ábra alapján állítottuk be.



19. ábra: 2. feladat – a KEPVISELO egyedtípus KNEV, KTEL és KJUT tulajdonságai mezőtulajdonságainak beállításai

A képviselő jutalékánál beállítottuk a feladatban írt érvényességi szabályt. A szabály alatt lehetőségünk nyílik érvényesítési szöveg megadására is. Ez akkor jelenik meg, ha a felhasználó rossz értéket adott meg, így egyénivé tehetjük az Access beépített sablon-hibaüzenetét (lásd alábbi ábra). Viszont figyeljünk arra, hogy az egyéni üzenet informatív legyen, nem elég pl. a „Rossz érték!” felhívás, erre a felhasználó már a felugró ablakból is rájön, de nem fogja tudni, mi a jó érték.

A KJUT mezőhöz bájt altípus került beállításra, az érvényességi szabályon belüli értékek e tárolási méretbe belefférnek.

20. ábra: 2. feladat – egyéni hibaüzenet

Hozzuk létre a HORECA adattáblát is (*Létrehozás/Tábla*), majd mentsük el és nevezzük át, aztán a tervező nézetbe átlépve vegyük fel a szükséges mezőket, állítsuk be az elsődleges kulcsot!

Az alábbiakban rövid összegzést adunk az egyes tulajdonságtípusok mezőtulajdonságainak beállításáról. A feladat alapján a szerződés azonosítóját a rendszer generálja, így a számláló adattípus került beállításra. A dátumoknál a rövid dátum formátum és a megfelelő beviteli maszk is kiválasztásra került. A HCEGDAT tulajdonságtípus érvényességi szabálya esetén az egyszerűbb megoldás az alábbi kifejezés használata:

```
<=Date()-730
```

A 730 kétszer 365-öt jelent. Ezt a kivonást azért tehetjük meg, mert a dátumkezelés az Excelhez hasonlóan a napok számára épül. A dátumforma mögött egy-egy konkrét szám áll, minden következő nap eggyel nagyobb értéket jelent. Tehát a mai dátumból 730-at levonva a 730 nappal korábbi dátumot kapjuk meg. Az egyetlen probléma a szökőévek kapcsán jelentkezik, ha a szökőnap belesik a kérdéses két évbe, a szabály a két évnél egy nappal fiatalabb cégeket is lefogadja.

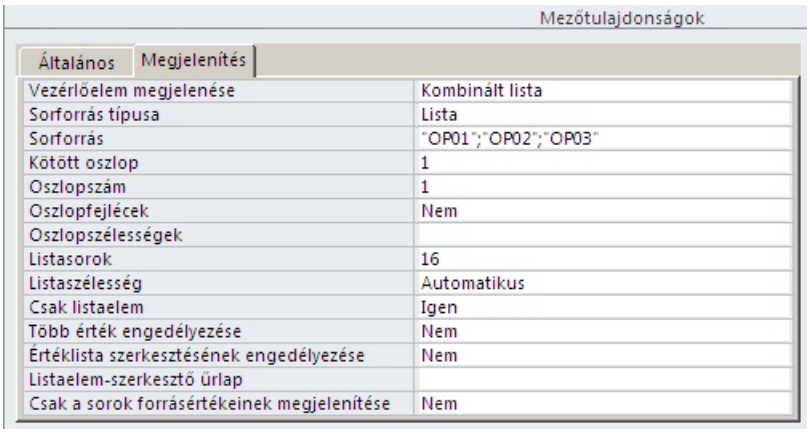
A pontosabb megoldás:

```
<=CDate(Year(Date())-2 & "." & Month(Date()) & "." & Day(Date()) & ".")
```

Vesszük a dátum év, hónap és nap részét, az év részből levonunk kettőt, majd összefűzzük a részeket és dátummá konvertáljuk.

A Date() függvényt használtuk a korábban megismert Now() helyett. A Now() az aktuális dátummal és idővel tér vissza, a pontos időre viszont nincs szükségünk. Viszont a fenti beállítások esetén, ha a Date() függvényt mindenhola Now() függvénnyel helyettesítjük, a megoldás akkor is tökéletesen működik, mivel a rövid dátum formátum és a formátummaszk beállítása levágja a Now() függvény által visszaadott időpontot, csak a visszaadott dátumrész marad meg. A kifejezésszerkesztő alsó részén megjelenő rövid leírás felett a kijelölt függvény neve is megjelenik, melyre kattintva részletes súgó érhető el az egyes függvényekről.

A szerződés extra opciója (HOP) tulajdonságtípus csak három alternatívát fogadhat be. Ezt a mezőtulajdonságok megjelenítés fülén állíthatjuk be. A vezérlőelem megjelenése legyen kombinált lista, amely egy legördülő menüt jelent. Ennél az elemnél lehetőségünk nyílik előre definiált listaelemekre korlátozni a rögzíthető értékeket. Az értékeket a sorforrás sorban adhatjuk meg. A szöveg típusú értékeket idézőjelbe kell tennünk, a listaelemek közé pontosvessző kerül. Előtte a sorforrás típusában ki kell választanunk a lista típust. Látható, hogy akár egy táblázat, vagy egy lekérdezés eredménye is lehetne az adott mező értékeinek korlátja.

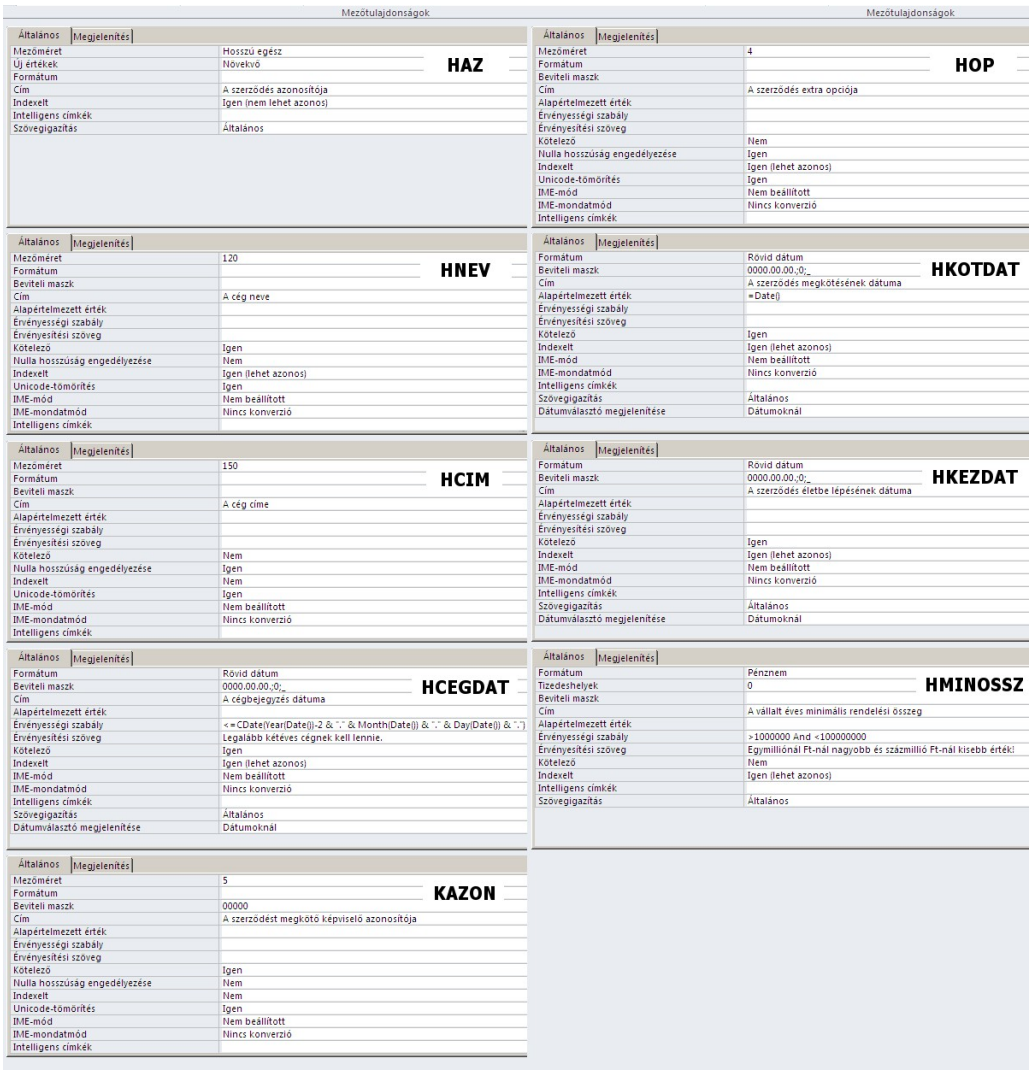


21. ábra: 2. feladat – a mező lehetséges értékeinek korlátozása

A HKOTDAT alapértelmezett értékénél szintén a Date() függvényt használtuk fel.

A további beállítások bizonyos fokig opcionálisak. Nem mondhatjuk, hogy csak egy út létezik, viszont az is biztos, hogy egyes beállítások rosszak. Például eldönthetjük, hogy a HCEGDAT tulajdonságtípust szeretnénk-e indexelni, vagy sem. Viszont az igen (nem lehet azonos) beállítás biztosan rossz, hiszen ezzel kizárjuk, hogy két vállalkozás egyazon napon került volna bejegyzésre. A képviselő azonosítóját tartalmazó KAZON tulajdonságtípust biztosan indexelni kell, hiszen ez adja meg az összeköttetést a KEPVISELO táblával. Viszont itt a nem lehet azonos beállítás szintén nem megfelelő, hiszen az azonosító itt idegen kulcsként szerepel, egy képviselő pedig több szerződést is köthetett. Ez a korábban tárgyalt egy-a-többhöz kapcsolat alapja. Kérdés, hogy a szerződést kötő vállalkozás (HNEV) szerepelhet-e többször. Miért is (ne) kötnénk egy céggel több szerződést? Nehéz válaszolni erre az 1. feladatnál is tárgyalt kérdésre. Az adatbázis lehetőségeit érdemes pontosan definiálni, viszont jelen esetben, úgy döntöttünk, hogy nem zárjuk ki a többszöri előfordulás lehetőségét, ezért az igen (lehet azonos) indexelést választottuk. Ettől még nem kötelező, hogy vállalkozások többször szerepeljenek. Ha több év szerepelhet egyszerre az adatbázisban, akkor a többszöri előfordulás logikus. A döntést indokolja az a tény is, hogy éles adatbázisban nem szokás törölni. Általában egy külön tulajdonság jelzi, hogy egy rekord aktív, vagy sem. Így a megjelenítés szempontjából töröltnek tűnik egy-egy tétel, valójában azonban továbbra is az adatbázisban marad, szükség esetén visszakereshetőek a tételek.

Az egyes mezők általunk beállított mezőtulajdonságait az alábbi ábra mutatja.



22. ábra: 2. feladat – mezőbeállások a HORECA egyedtípus tulajdonságaiban

Két feltételnél problémáink akadnak. A szerződés életbe lépésének dátuma (HKEZDAT) nagyobb-egyenlő, mint a szerződéskötés dátuma. Ahhoz, hogy ezt a feltételt be tudjuk állítani, tudnunk kellene a HKOTDAT mezőre hivatkozni, a tábla érvényesítési szabályának kifejezésszerkesztője viszont nem hozza fel ezt a lehetőséget, itt nem nyílik mód erre. Ugyanez a probléma a HMINOSSZ oszthatósági feltételével, ahol szintén a saját mezőre történő hivatkozás lehetőségére van (volna) szükség a beállításhoz.

Nyissuk meg a HORECA táblát adatlap nézetben, majd menjünk a *Táblaeszközök/Mezők/Érvényesítés* gombhoz. Itt lehetőségünk nyílik rekordérvényesítési szabályok beállítására is. Ez az adott rekord felvitelének a végén, a továbblépéskor értékelődik ki – tehát a kérdéses cellákba bármit beírhatunk, a cellából továbblépéskor még nem fogja jelezni a rendszer a feltétel megsértését, ellentétben a mezőérvényességi szabállyal. A rekordérvényesítési szabály kifejezésszerkesztőjének megnyitásakor látható, hogy itt már a HORECA tábla azonosítóival is dolgozhatunk. Adjunk meg olyan logikai kifejezést, amely igaz értéket ad vissza, ha az általunk elvárt szabályoknak megfelel a rekord, és hamisat, ha nem. Az egyik feltétel tehát, hogy a HKEZDAT értéke nagyobb-egyenlő, mint a HKOTDAT értéke.

A másik feltételünk talán fogósabb. A vállalt minimumértéknek (HMINOSSZ) százezerrel oszthatónak kell lennie. Ilyen függvény nincs, de könnyen kialakíthatunk e vizsgálathoz megfelelő egyenlőséget. Ha a konkrét értéket elosztjuk százezerrel, a feltétel szempontjából két kimenet lehetséges. Egész értéket kapunk, ha fennáll az oszthatóság, törtet, ha nem. Ha tehát a hányados egészrésze megegyezik

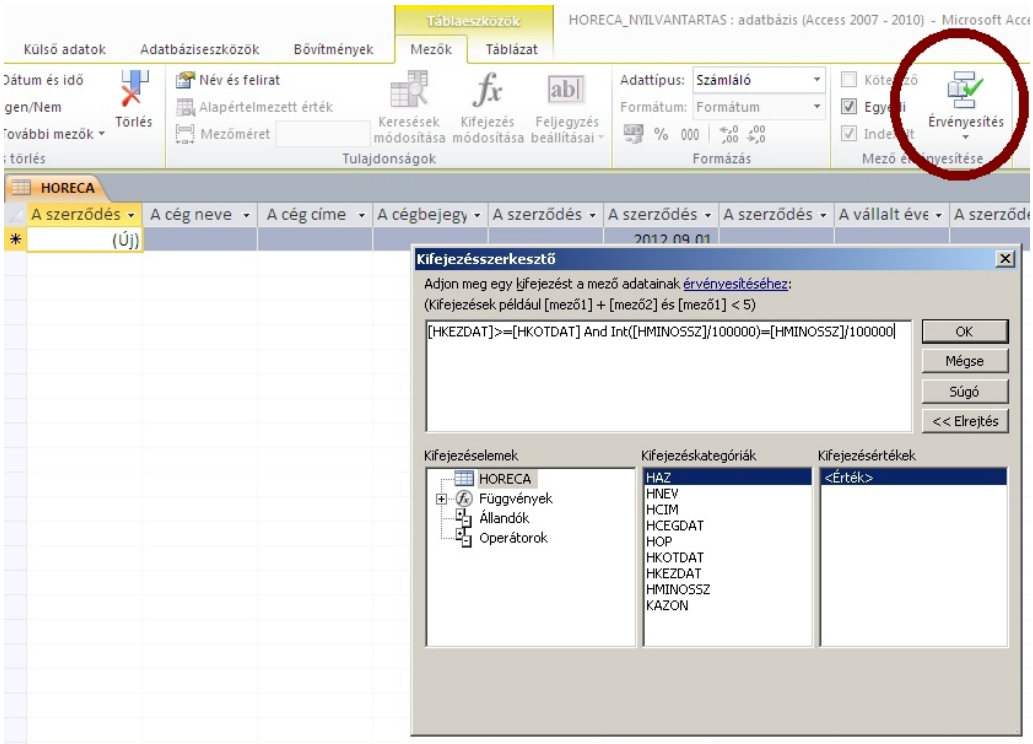
magával a hányadossal, akkor teljesül a feltételünk. Ehhez az egészrész függvényt (Int) fogjuk felhasználni. A feltétel:

```
Int([HMINOSSZ]/100000)=[HMINOSSZ]/100000
```

A két feltétel mindegyikének teljesülnie kell egy-egy rekordban, ezért egy *And* operátorral kapcsoljuk őket össze.

```
[HKEZDAT]>=[HKOTDAT] And Int([HMINOSSZ]/100000)=[HMINOSSZ]/100000
```

Itt is lehetőségünk nyílik egyéni érvényesítési üzenet megjelenítésére, mely *Táblaeszközök/Mezők/Érvényesítés* gomb alatt állítható be. A szövegben utaljunk az összes megkötésre: „A szerződés életbe lépésének dátuma nem lehet korábbi a szerződéskötésnél, továbbá a vállalt minimális összegnek százezerrel oszthatónak kell lennie.”

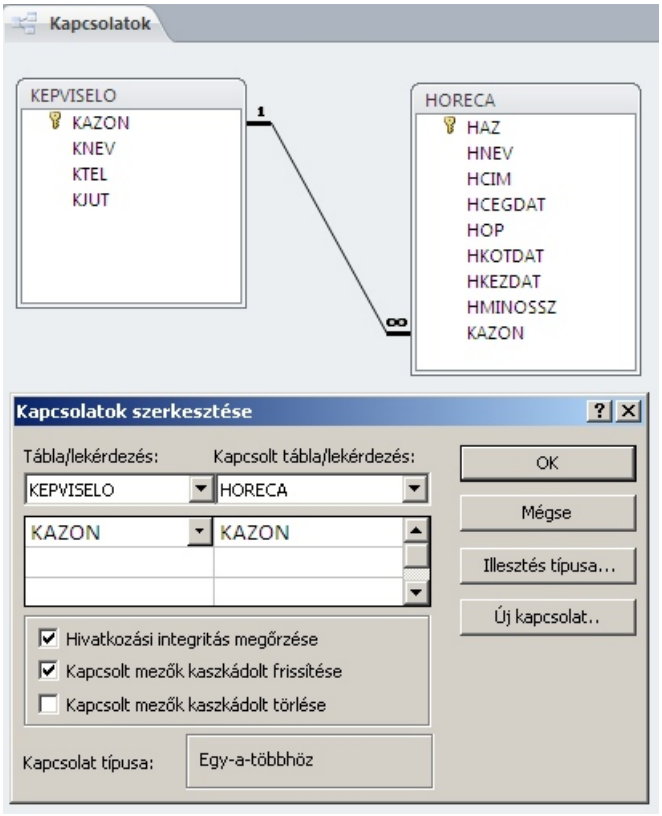


23. ábra: 2. feladat – rekordérvényesítés

A feladat utolsó lépésében hozzuk létre a táblák közötti kapcsolatot a korábban ismertetett módon (*Adatbáziseszközök/Kapcsolatok*).

A feladat nem adja meg, hogy mit kezdjünk a hivatkozási integritás nyújtotta lehetőségekkel, de ez nem jelenti azt, hogy ne kellene döntenünk. Hasznos lehet, ha a HORECA táblába nem tudunk olyan sorokat rögzíteni, ahol a kapcsolatot biztosító azonosító a KEPVISELO táblában nem létezik. Csak létező személy köthet szerződést, a vállalkozástól kilépő képviselőket pedig nem is célunk törölni, maximum archiváljuk a nem aktív rekordokat. A hivatkozási integritás megőrzését tehát célszerű kiválasztani. Egy konkrét képviselő törlési lehetőségének biztosításakor a gyakorlati életben általában nem törlünk az adatbázisból, csak egy az archiválás-jelző mezőt állítunk igazra. Itt most nincs megadva ez a mező, de képzeljük hozzá az adatbázisunkhoz. Mivel a törlést csak ilyen értelemben fogjuk biztosítani, ezért az (össze)kapcsolt mezők kaszkádolt törlésére nincsen szükség. Ha ezt nem engedjük, úgy, ha mégis valamilyen törlési művelet kerülne kiadásra olyan sorra, amelyhez tartoznak rekordok a HORECA táblában, hibaüzenetet kapnánk, nem pedig egy háttérben lefutó folyamatot, amely a HORECA táblában is törléseket végez. A kapcsolt mezők kaszkádolt frissítése a KEPVISELO táblában található elsődleges kulcsok módosításakor lehet hasznos. Mint korábban említettük, a felhasználók az adatrögzítést űrlapokon keresztül végzik. Ha az elsődleges kulcs a rendszer által biztosított számláló, akkor az űrlapon nem is biztosítunk lehetőséget ennek módosítására, így a kapcsolt mezők kaszkádolt

frissítésének sincs haszna. Jelen esetben tegyük fel, hogy elképzelhető az egyes képviselők azonosítójának változtatása, ezért a kaszkádolt frissítést kapcsoljuk be!



24. ábra: 2. feladat – tábla kapcsolatok

Az Access további eszközöket is kínál, melyek egy része nem is kötődik szorosan a szűken értelmezett adatbázis-kezeléshez. Javasoljuk, hogy a kedves olvasó próbálja ki például a kimutatás nézetet is – majd a 4. feladat, az adatfeltöltés után –, mi erre itt nem térünk ki.

3. feladat – több mezőből álló indexek

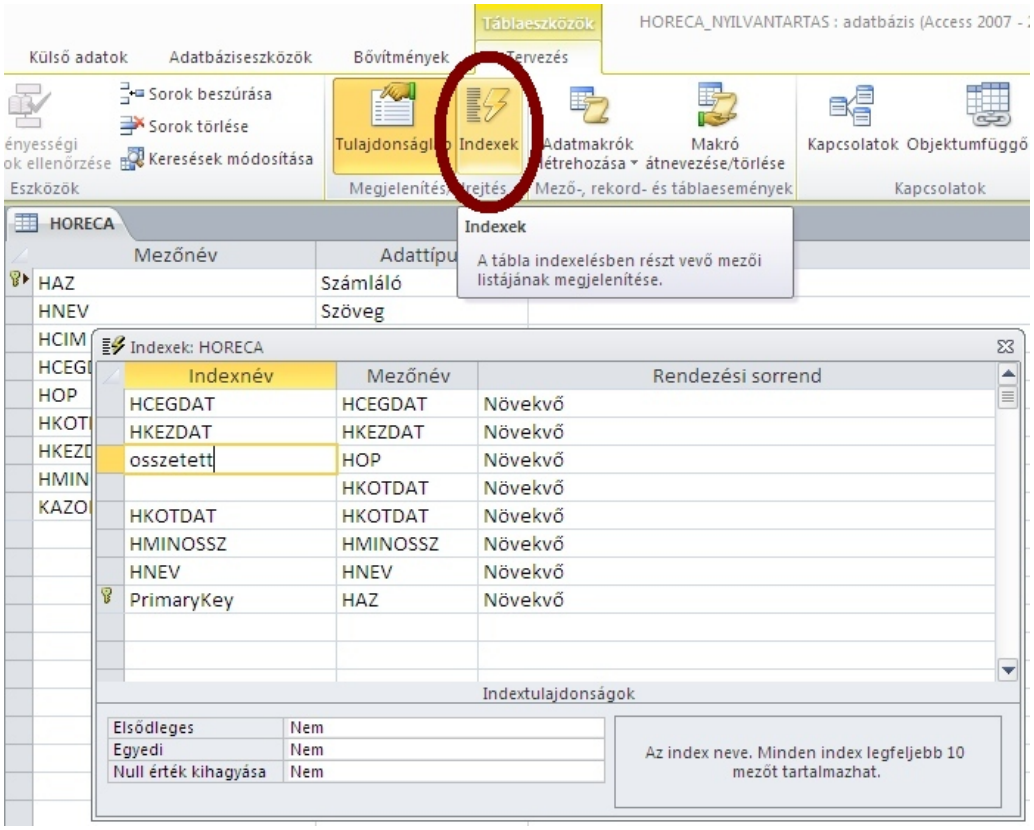
Hozzon létre több mezőből álló indexet a HORECA tábla HOP és HKOTDAT tulajdonságaira alapozva!

Tegyük fel, hogy gyakran végzünk kereséseket több mező alapján. Ebben az esetben ezekhez érdemes önálló indexeket létrehozni, melyek a különböző mezőkombinációkat indexelik. Több mezőből álló index esetén az Access először az index első mezője szerint rendez. A mezők sorrendjét a több mezőből álló index létrehozásakor állítjuk be. Ha több olyan rekord is van, amely az első mezőben azonos értékkel rendelkezik, akkor az Access az index második mezője szerint is elvégzi a rendezést, és így tovább.

A kérdésben azzal a feltételezéssel élünk, hogy gyakran kérdezünk le az extra opciókra (HOP) és a kötési dátumokra (HKOTDAT) vonatkozóan, tehát a két mező alapján együttesen. Például, hogy milyen szerződéseink vannak X és Y kötési dátumhatárok között az OP01 opcióval. Ez esetben szükségünk van egy összetett indexre.

Nyissuk meg a HORECA adattáblát tervező nézetben, majd válasszuk a *Táblaeszközök/Tervezés/Indexek* menüpontot! A felugró ablakban az összes beállított indexet látjuk, ezeket is menedzselni tudjuk, valamint a rendezési sorrendek beállítására is lehetőség nyílik. Összetett, több mezőből álló index megadásakor csak az első mezőnévnél szerepelhet az indexnév. Egészen a következő indexnévig, a felsorakoztatott mezők az adott index részét képezik –max. 16 darab mezőnév kerülhet egy indexbe, ez a korlát bőven elég komolyabb adatbázisok esetén is. Az egyedi tulajdonságot nem kell állítanunk, hiszen a HOP, HKOTDAT értékkombinációk előfordulhatnak többször is. Az alábbi ábrán látható az

általunk „osszetett” néven létrehozott index. Hogy egyértelműek legyenek a határok, mi a létező indexek közé szúrtuk be, de legalulra is kerülhetett volna.



25. ábra: 3. feladat – több mezőből álló index létrehozása

4. feladat – adatfeltöltés Excelből

Töltse fel az adattáblákat.

Az adatfeltöltést, mint már említettük, a felhasználók az adatbázis kialakítása után űrlapokon keresztül fogják elérni. Az adatfeltöltés tehát az adatbázis kialakítása után következik, ott pedig még nem tartunk. Miért került ide ez a feladat? A feltöltés a tesztekhez szükséges, a különböző adatok a következő feladatok megfelelő megoldásának ellenőrzését segítik.

Lehetőség nyílik tömeges adatfeltöltésre, importálásra is, mi ezzel fogunk élni.

Töltsük le az ide linkelt Excel állományt. [images/HORECA_nyilv_adatfeltoltes_4feladat.xlsx]

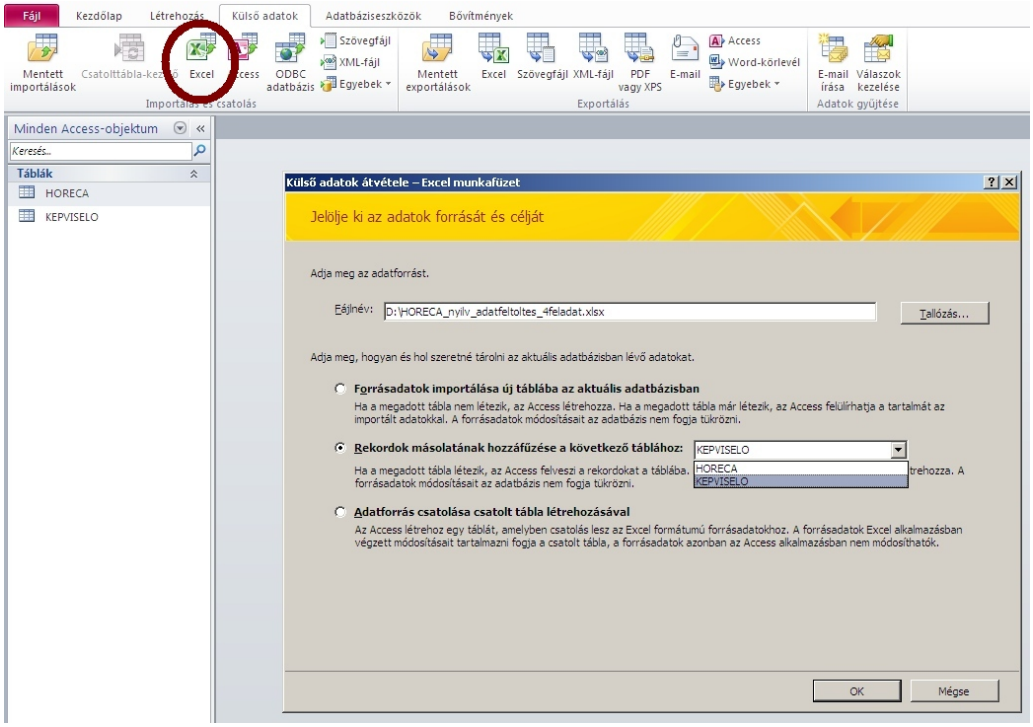
Ha megnyitjuk, láthatjuk a mintaadatokat, amelyeket két külön munkalapra rendeztünk. A munkalapok elnevezése igazodik a tábláinkéhoz. Az első adatsorokban a mezőnevek szerepelnek, majd a felviendő rekordok következnek. A megadott adatok betartják a táblák szabályait. Egyetlen kivétel látható, a HORECA táblába szánt adatok első két sora azonos kulccsal került rögzítésre, a második sor cégnevénél a törölve szó szerepel.

Zárjuk be az Excelt, valamint az Accessben megnyitott táblákat, majd kattintsunk a *Külső adatok* lap, *Importálás és csatolás* csoportjában található *Excel* gombra! A felugró ablakban be kell tallóznunk az Excel állományunkat, majd válasszuk a *rekord másolatának hozzáfűzése a következő táblához*: KEPVISELO parancsot. E táblával kezdjünk, hogy megfeleljünk a korábban beállított hivatkozási integritás kritériumának: nem kerülhet fel olyan szerződés, amelyhez nem kapcsolódik a KEPVISELO tábla egy eleme, ami viszont jelenleg üres. A felbukkanó táblázat importálása varázslóban nincs más dolgunk, mint kiválasztani a megfelelő munkalapot, és tovább lépni. A Befejezés gomb megnyomásával az adatokat importáljuk. A HORECA tábla feltöltéséhez az előző lépéseken kell újra végigmennünk.

A befejezés gomb lenyomásakor hibaüzenettel szembesülünk, az ismétlődő azonosítójú rekord nem kerül importálásra, hiszen az azonosító, mint elsődleges kulcs nem tartalmazhat egyező értékeket. Kattintsunk a hibaüzeneten a folytatást jelentő Igen gombra, így a többi adat bekerül a táblába.

Ha újabb sort próbálunk felvinni a HORECA táblába, a szerződés azonosítója automatikusan töltődik, hiszen számláló adattípussal vettük fel. Ha az üres táblát próbáltuk volna meg feltölteni, egyestől kezdődött volna a számozás, a feltöltés után viszont a meglévő hét számjegyű azonosítókat folytatja a rendszer.

Ne feledjük, hogy módosításkor az éles adatbázisba dolgozunk. A mögöttes logika eltér az irodai szoftvercsomag többi termékétől. Itt a módosítások közvetlenül érintik az adatbázist, ezért is ugrik elő oly sokszor a mentés. Az egyes műveletek véglegesítése, vagy elvetése után nyílik csak lehetőség a következő műveletre. Nem úgy, mint a Wordben vagy Excelben, ahol sok-sok művelet után vagy mentünk, vagy nem. Ez a megközelítés a visszavonást is érinti, sok műveletnél a véglegesítés – mentés – után már nincs lehetőség a visszavonás használatára. Erre érdemes figyelni. Ezen felül is sok korlátba ütközhetünk. Az egyes táblákban például sok korlátozást állítottunk be. A rendszer addig nem enged továbblépni az adott rekordról, míg minden szabálynak meg nem felelünk. Ha menet közben mégis kilépnénk, csak a hibaüzeneteket kapjuk. Ilyenkor az Esc gomb – a billentyűzeten – jó barátunk, segít visszavonni az aktuális műveleteket.



26. ábra: 4. feladat – adatfeltöltés Excelből

5. feladat – választó lekérdezés

Tervezzen olyan OP01 nevű lekérdezést, amely az OP01 extra opcióval rendelkező szerződő cégek nevét, címét, valamint vállalt minimális rendelési összegüket jeleníti meg a szerződés életbe lépésének dátumára csökkenően! Az oszlopfeliratok legyenek ékezetesek, nagybetűvel kezdődőek!

Első lépésben gondoljuk át, melyik táblákból van szükségünk adatokra! Meg kell vizsgálni az extra opciót (HOP), kell a cégnév (HNEV), a cím (HCIM), a vállalt minimális rendelési összeg (HMINOSSZ) és a szerződés életbe lépésének dátuma (HKEZDAT). Ez mind a HORECA táblában található. Válasszuk a *Létrehozás/Lekérdezéstervező* gombot a menüben! A menüsorban automatikusan a *Lekérdezőeszközök/Tervező* lap látszik, látható, hogy alapértelmezésben választó lekérdezést nyitottunk meg. A felbukkanó ablakban csak a HORECA táblát kell a lekérdezésünkhöz adnunk. Miután már átgondoltuk mely mezőkre van szükségünk, az alsó tervezőrácson állítsuk be ezeket a *mező* sorban.

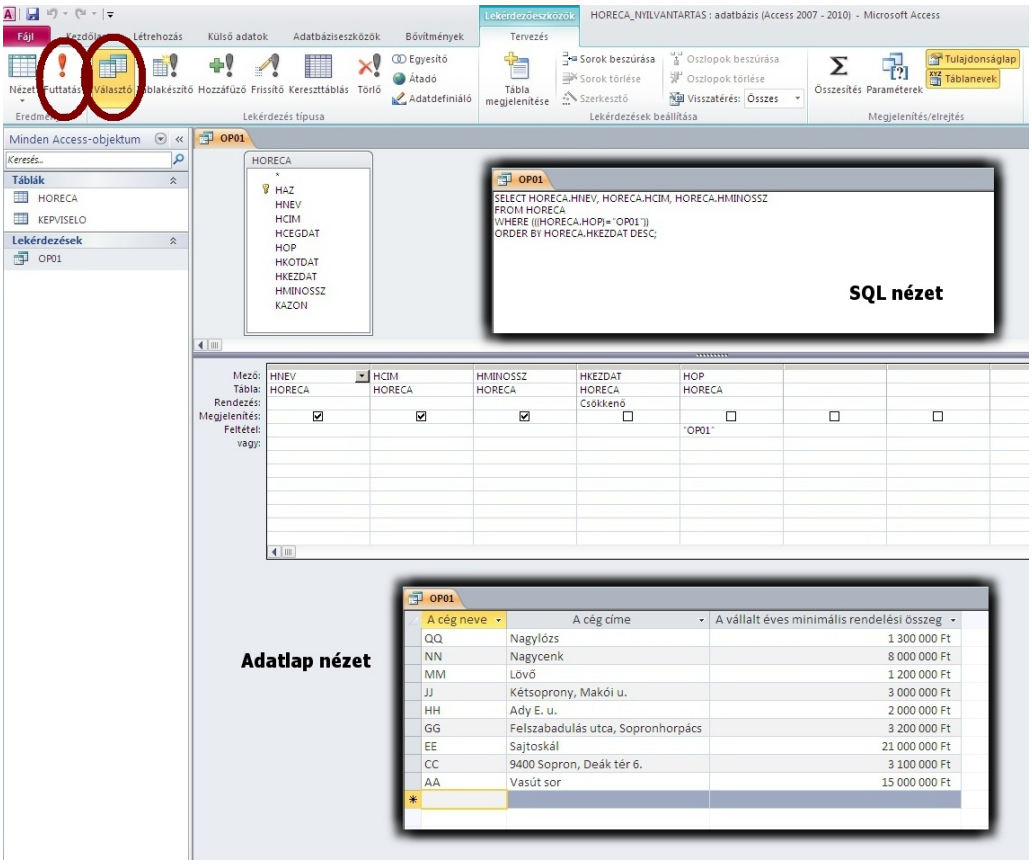
(A * az adott tábla összes mezőjét jelenti.) A megjelenítés sorban kapcsoljuk le a HOP és HKEZDAT mezők megjelenítését, ezekre a válogatási és rendezési feltételek megadása végett van szükségünk, a lekérdezett adatok között nem szeretnénk látni ezeket az információkat. Logikus is, pl. a HOP minden eredménye OP01 lesz, ezt már most is tudjuk, nincs szükségünk erre az eredményoszlopra. A HOP mezőnek *="OP01"* lesz a feltétele, az ilyen extrával rendelkező szerződések adatait szeretnénk leválogatni. A kezdő dátumnál a rendezést kell csökkenőre állítani.

Lépjünk át adatlap nézetbe (ezt itt az eszköztáron található futtatás gombra kattintva is megtehetjük), láthatjuk a lekérdezés eredményét. Lépjünk vissza a tervező nézetbe, állítsuk a HOP és HKEZDAT mezők megjelenítését is igazra (pipa), majd lépjünk vissza az adatlap nézetbe és ellenőrizzük: csak OP01 extrák láthatóak, a sorok pedig a szerződés életbe lépési dátumának csökkenő sorrendjében jelennek meg. Az ellenőrzés után állítsuk vissza a megjelenítések feladatban kért beállításokat! A lekérdezés bezárásakor (elérhető pl. a lekérdezés fülén jobb egérgombbal előhívható helyi menüből) automatikusan felajánlja a rendszer a mentést, mentsük el a lekérdezést a feladatban megadott néven.

A lekérdezések grafikus elemei mögött az SQL lekérdező nyelv áll. A nézetek között lehetőségünk van SQL nézetbe átlépni, tulajdonképpen a teljes lekérdezést megírhattuk volna SQL-ben. Ha átlépünk, a következőt látjuk:

```
SELECT HORECA.HNEV, HORECA.HCIM, HORECA.HMINOSSZ
FROM HORECA
WHERE (((HORECA.HOP)="OP01"))
ORDER BY HORECA.HKEZDAT DESC;
```

A SELECT utasításban határozzuk meg, hogy milyen mezőket szeretnénk látni a lekérdezés eredményében Táblanév.Mezőnév azonosítással. Az Access automatikusan a mezőnév cím mezőtulajdonságában megadott értéket szerepelteti az oszlop fejlécében. A FROM záradék adja meg a használt táblákat, a WHERE záradék pedig a kiválasztási feltételeket és a táblák közötti kapcsolat feltételeit. Most a HORECA.HOP mezőnek kell egyenlőnek lennie azOP01 értékkel, amely szöveges adattípusként idézőjelek közé kerül. Az Access által beszúrt mindhárom pár zárójel felesleges, törölhető, de a kiértékelést nem zavarják. Az ORBER BY záradék a rendezést definiálja, ez akár többszintű is lehet. A lekérdezést pontosvessző zárja le. Ha a DESC szót átírjuk ASC-re, az OP1-et, pedig OP2-re, majd visszalépünk tervező nézetbe, akkor a csökkenő feltétel növekvőre váltott, az =”OP1” feltétel pedig =”OP2”-re. Az SQL utasítások módosítása tehát a grafikus felületet is átalakítja és fordítva.



27. ábra: 5. feladat – a választó lekérdezés nézetei

6. feladat – paraméteres lekérdezés

Tervezzen Havidíj néven olyan paraméteres lekérdezést, amely a paraméterként megadott hónapra megjeleníti a képviselők azonosítóját, nevét, az adott hónapban általuk kötött szerződésekben foglalt vállalt minimális rendelések összegét, valamint a képviselők havi jutalékát!

Az adatbázist mi tervezőként, az adatbázis előállítójaként látjuk. A felhasználók az előre definiált lekérdezések futtatására lesznek jogosultak. Jelen állapotban összesen egy lekérdezési lehetőségük lenne, amit az előző feladatban létrehoztunk. Az adattáblák szerkezetének felépítése után most további lekérdezéseket hozunk létre. Inkább a lehetőségek bemutatására törekszenek a feladatok, nem készítünk például számos különböző választó lekérdezést, ami a megfelelő funkcionalitáshoz elengedhetetlen lenne. A felhasználók az adatbázis különböző szempontok szerinti lekérdezési lehetőségeit általában felajánlott gombsorokon keresztül érik el – gondoljunk pl. egy weboldalra. Sokszor előfordulhat azonban, hogy maga a lekérdezés csak egy keret, bizonyos szükséges információkat nem tudunk előre meghatározni, azokat paraméterként futtatáskor kérjük be. Ehhez nyújt segítséget a paraméteres lekérdezés, mely tulajdonképpen szintén egy választó lekérdezés.

A szükséges adatok a képviselő azonosítója (KAZON), neve (KNEV), a szerződéses vállalások (HMINOSSZ), a szerződéskötések dátuma (HKOTDAT), valamint a képviselő jutaléka (KJUT). Ezekből megállapíthatjuk, hogy mindkét táblánkra szükségünk lesz. A *Létrehozás/Lekérdezéstervező* gomb segítségével hozzunk létre egy új lekérdezést, majd adjuk hozzá a tábláinkat! Adjuk hozzá a KAZON, KNEV, és HMINOSSZ tulajdonságokat! A KAZON mező esetében felmerül, hogy melyik táblából. A KEPVISELO táblából válasszuk, ott ez egyértelműen a képviselőhöz rendelt elsődleges kulcs.

A mostani feladat nem csak a paraméter megadás lehetőségében tér el az előzőtől. Most nem az adatbázis soraira vagyunk kíváncsiak, tehát nem az a cél, hogy kilistázzuk, hogy X azonosítóval Y képviselő egymás alatti Z darab sorban szerepel a kérdéses hónapban a kért adatokkal. Azt szeretnénk

látni, hogy adott azonosítójú képviselő a kérdéses hónapban összesen mekkora összegre kötött szerződést. Tehát egy képviselő egyszer jelenik meg, egyetlen sorban és mellette az általa adott hónapban kötött szerződések minimális rendelési kötelezettségeinek összege szerepel. Az összegen van a hangsúly. Nem sorokat/rekordokat listázunk, hanem sorokat csoportosítunk, mezőket összegzünk. A csoportosítás alapja elsődlegesen a képviselő kódja, másodlagosan a kódhoz tartozó név. Csak a név nem lenne elég, azonos nevű képviselők esetén az ő összegeik is összeadódnának. A minimális rendelési összegeket pedig összegezni kell.

A *Lekérdezőeszközök/Tervezés/Összesítés* gomb egy új, *összesítés* sorral bővíti a *Tervezőrácsot*. Ebben különböző függvények[1] mellett a Group By, Where és Expression kitételek szerepelnek. A Group By azokra az elemekre vonatkozik, amelyek csoportosítása mentén szeretnénk valamilyen számításokat elvégezni. Ez az alapértelmezett érték, ha nincs más beállítva, nincs számítás, akkor tulajdonképpen a feltételeknek megfelelő rekordok listáját jelenítjük meg – lásd előző feladat. Most a KEPVISELO.KAZON és KEPVISELO.NEV azonosítók a csoportosító elemek. A HORECA.HMINOSSZ összegeit szeretnénk látni, itt válasszuk a Sum lehetőséget az összesítés sorában. (Még a Sum kiválasztása előtt meg is nézhetjük adatlap nézetben az eredményt, majd a Sum beállítása után újra. Így biztosan egyértelmű lesz, mit is jelent az összegzés.)

A jutalékokat nem egyszerűen összegezni szeretnénk, nem is lenne értelme. Ha pl. egy képviselő 12 ezrelék jutalékot kap, és összesen hat szerződést kötött, akkor a neve mellett Sum beállítás esetén a jutalék oszlopban 60 állna. Nekünk a jutalék kiszámítása lenne fontos: *minimális szerződési összeg * képviselő jutaléka / 1000* (az ezrelék miatt). És ezeket szeretnénk összegezni. Ehhez egy kifejezést (Expression) kell írnunk:

```
Sum(KJUT*HMINOSSZ/1000)
```

A Sum is szükséges, hiszen a szorzás csak az adott rekordokra számítja ki a jutalékot, melyet a csoportosító tényezők szintjéig összegeznünk is kell, ezt viszont most nem tudjuk az összesítés oszlopban beállítani. Írjuk be a következő üres oszlop mezők cellájába a fenti kifejezést, az összesítésnél pedig válasszuk az Expression kitételt. Futtatással ellenőrizhetjük az eredményt. A megadott különböző nevek kapcsos zárójelbe kerülnek, ezeket az Access automatikusan összeköti a létező objektumok-tulajdonságokkal. Ha ez nem sikerül, akkor bekérendő paraméterként kezeli. Ha így jártunk, ellenőrizzük, nem gépeltünk-e el valamit.

A kifejezés előtt megjelent a *Kif1:* felirat, adatlap nézetben pedig ez lett az adott mező szöveges fejléce. A HMINOSSZ oszlop fejléce is változott. Mostani tartalma nem egyezik az egyes szerződések vállalt minimális rendelési értékével, ezek képviselők szerinti összegzését takarja. Ezért az eredeti tulajdonság megnevezése nem alkalmazható, a rendszer a *SumOf HMINOSSZ* fejlécet adta a számított oszlopnak. Ha szeretnénk egyéni szöveget, úgy kell eljárnunk, mint a fenti kifejezésnél. A mező tulajdonság elejére írjuk be a fejléc egyéni szövegét, majd kettőspont. A kettőspont, mint elválasztó elem előtti rész az oszlopfejléc szövegébe kerül. Ez valamennyi oszlopra igaz. Írjuk a HMINOSSZ elé, hogy V*állalt minimális rendelési értékek összegzése:* . Ez lesz az oszlop fejléce. A Kif1: -ot írjuk át a *Képviselő jutaléka:* szövegre.

Most már csak azt kell beállítanunk, hogy a lekérdezésbe csak a felhasználó által választott hónap adatai kerüljenek be. Szükségünk lesz egy [hónap] paraméterre, amelybe futásidőben betöltődik a felhasználó által választott hónap. Ezt a *Lekérdezőeszközök/Tervezés/Paraméterek* gomb lenyomása után állíthatjuk be. A paraméter neve legyen hónap – a kapcsos zárójel az Access szintaktikájához tartozik – az adattípusa pedig a szám altípusának számító Bájt. Felmerülhet a kérdés, hogy mi történik, ha a felhasználó szövegesen próbálja megadni a kért hónapot. Ezt éles rendszer esetén kezelni kellene, mi most eltekintünk tőle. A hónap név helyett hosszabb kifejezést is megadhatnánk, hiszen a megadott név jelenik meg kérdésként a felhasználó előtt. Pl.: [Adja meg a választott hónap sorszámát:] .

Azokat a rekordokat szeretnénk a lekérdezésben tudni, amelyeknél hónap paraméterbe bekért érték egyezik a HKOTDAT hónap értékével. Mivel utóbbiban teljes dátum van, ki kell nyernünk a hónap értéket. Erre a korábban megismert Month() függvény alkalmas_

[1] Sum – összegzés, Avg – átlag, Min – minimum, Max – maximum, Count – darabszám, StDev – szórás, Var – szórásnégyzet, First – első elem, Last – utolsó elem

```
[hónap]=Month([HORECA].[HKOTDAT])
```

Vigyük fel a tervezőrács következő üres oszlopának mező értékébe a hónap szót, a kapcsos zárójelet a rendszer kirakja. Figyeljünk a pontos gépelésre, ha a felvett érték nem egyezik *A lekérdezés paraméterei* ablakban felvett értékkel, úgy két külön bekérendő adatként kezeli a rendszer. Az összesítésben válasszuk a Where lehetőséget, ezzel jelezzük, hogy itt egy feltételt fogalmaztunk meg. Az oszlop megjelenítését nem kérjük, a feltétel pedig *Month(HORECA.HKOTDAT)* . Az IntelliSense a bevitelkor automatikusan felajánlja a lehetőségeket. A kapcsos zárójeleket nem nekünk kell beírni. Az egyenlőségjel alapértelmezett, ezért nem szükséges kitenni (ha nincs relációs jel, a rendszer az egyenlőségjelet helyettesíti oda). A Kif1: -nek itt nincs szerepe, hiszen az oszlop nem jelenik meg, de átírható, pl. *A választott hónap:* -ra.

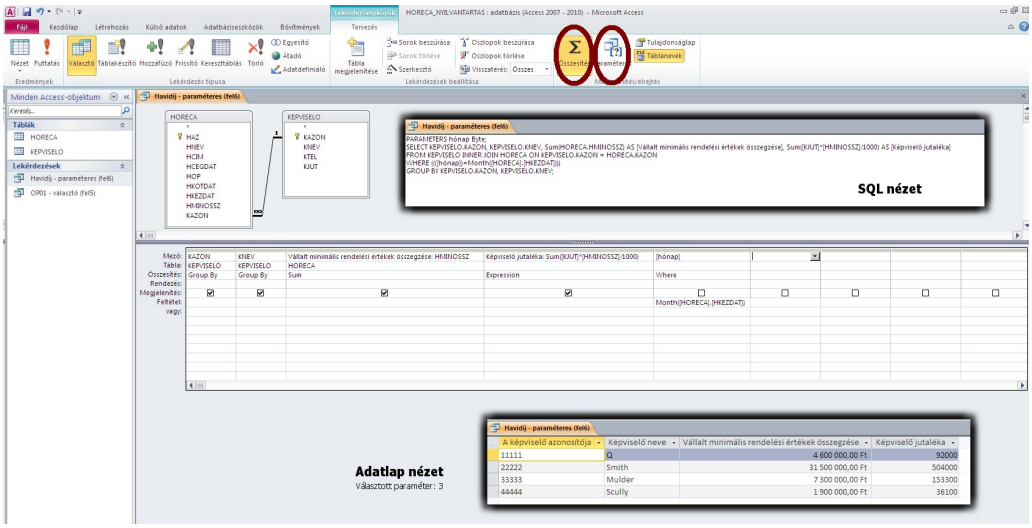
Kész is vagyunk, mentsük el, illetve teszteljük a lekérdezést. A 2,3 és 4 hónapértékekkel érdemes, a korábban felvitt adataink ilyen értékeket tartalmaznak.

Nézzük az SQL nézetet:

```
PARAMETERS hónap Byte;
SELECT KEPVISELO.KAZON, KEPVISELO.KNEV,
Sum(HORECA.HMINOSSZ) AS [Vállalt minimális rendelési értékek összegzése],
Sum([KJUT]*[HMINOSSZ]/1000) AS [Képviselő jutaléka]
FROM KEPVISELO INNER JOIN HORECA ON KEPVISELO.KAZON = HORECA.KAZON
WHERE ((([hónap])=Month([HORECA].[HKEZDAT])))
GROUP BY KEPVISELO.KAZON, KEPVISELO.KNEV;
```

Az első sorban a paraméterek definiálása történik meg. A SELECT utasításban az AS opció utáni kapcsos zárójel tartalmával felülírhatjuk az oszlopok fejlécét. Ha az egyéni oszlopfejléc egy szóból áll, a kapcsos zárójel elhagyható. A GROUP BY záradék lényegét korábban tárgyaltuk. Az előző feladatban azért nem volt rá szükség, mivel minden oszlop ide tartozott, nem voltak sorok összevonásával számított mezők. Az előző feladathoz képest a FROM záradék tartalma újdonság. Itt most két tábla szerepel, melyek kapcsán itt semmit nem állítottunk. Kapcsolatuk a kapcsolatok korábbi beállításából származik, melyet a táblák közötti kapcsolódást jelképező nyílra kattintva felül is írhattunk volna. Itt annyit emelnénk ki, hogy a 2. feladatban definiált kapcsolatok általános meghatározása nem minden esetben történik meg. Ilyenkor a kapcsolatokat az egyes lekérdezésekben külön definiálják. Ez itt úgy is történhetne, hogy a tervezőrácson egy külön oszlopban beállítanánk a KEPVISELO.KAZON és HORECA.KAZON egyenlőségét, Where „összekötéssel”. Az SQL a következőképpen alakulna (ez egyenértékű az előzővel):

```
PARAMETERS hónap Byte;
SELECT KEPVISELO.KAZON, KEPVISELO.KNEV,
Sum(HORECA.HMINOSSZ) AS [Vállalt minimális rendelési értékek összegzése],
Sum([KJUT]*[HMINOSSZ]/1000) AS [Képviselő jutaléka]
FROM HORECA, KEPVISELO
WHERE ((([hónap])=Month([HORECA].[HKEZDAT]))
AND ((KEPVISELO.KAZON)=[HORECA].[KAZON]))
GROUP BY KEPVISELO.KAZON, KEPVISELO.KNEV;
```



28. ábra: 6. feladat – a paraméteres lekérdezés nézetei

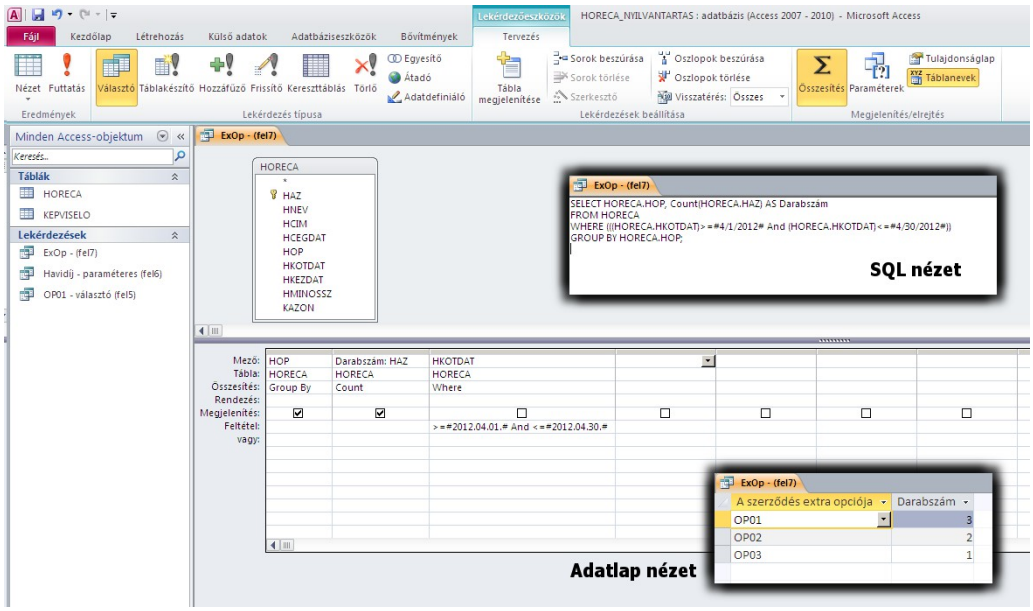
7. feladat – „összesítő” lekérdezés

Tervezzen olyan ExOp nevű lekérdezést, amely meghatározza a különböző extra opciókkal rendelkező, 2012 áprilisában megkötött szerződések számát!

Ez a feladat az előzőekben megismert lehetőségeket gyakoroltatja. A lekérdezésbe beletartozó rekordok határoló dátumait általános esetben akár paraméterekkel is be lehetne kérni, érdemes ennek megoldását is megpróbálni.

A *Count()* függvényre lesz szükségünk. A darabszámot kötelező mezőn érdemes számolni, mi az azonosítót választottuk erre a feladatra. Akár szöveg adattípusú mező is alkalmas lehet, mivel nem az értékeket akarjuk összeadni, hanem a cellák darabszámát szeretnénk megszámolni. A kötelező mező azért fontos, mert így biztos, hogy minden rekordban van adat az adott mezőben. A rendszer az üres, kitöltetetlen előfordulásokat nem számolja. Ki is próbálhatjuk: a feladat megoldása után töröljük ki a HORECA tábla utolsó rekordjából a címet (Nagylózs), ez áprilisi hónapban kötött szerződést tartalmazó rekord, a cím megadása pedig nem kötelező. Az összegzést állítsuk át a HCIM mezőre. Az OP01 opcióhoz eggyel kisebb darabszám kerül, az üres értéket tényleg nem számolja a rendszer. (Ne felejtsük el visszaállítani a HORECA táblát.)

A feladat megoldását az alábbi ábrán szemléltetjük:



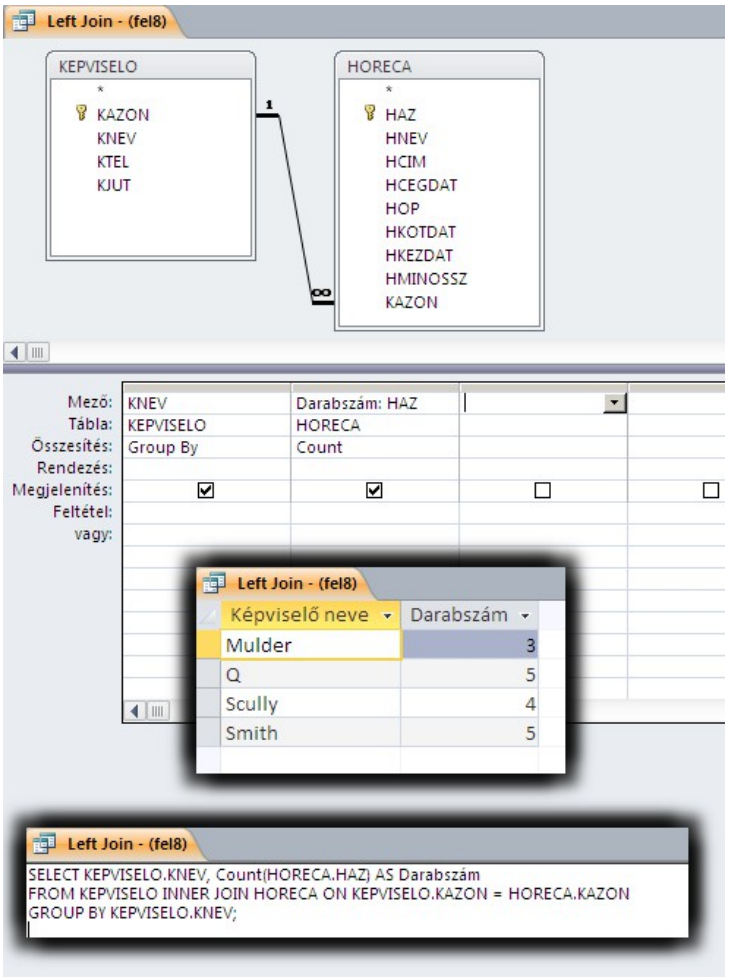
29. ábra: 7. feladat – a lekérdezés nézetei

8. feladat – Left join

Készítsen lekérdezést Left Join néven, amely az összes képviselő nevét tartalmazza, illetve az általuk kötött biztosítások számát!

Miben új ez a feladat? Készítsük el eddigi ismereteink alapján, aztán majd kiderül.

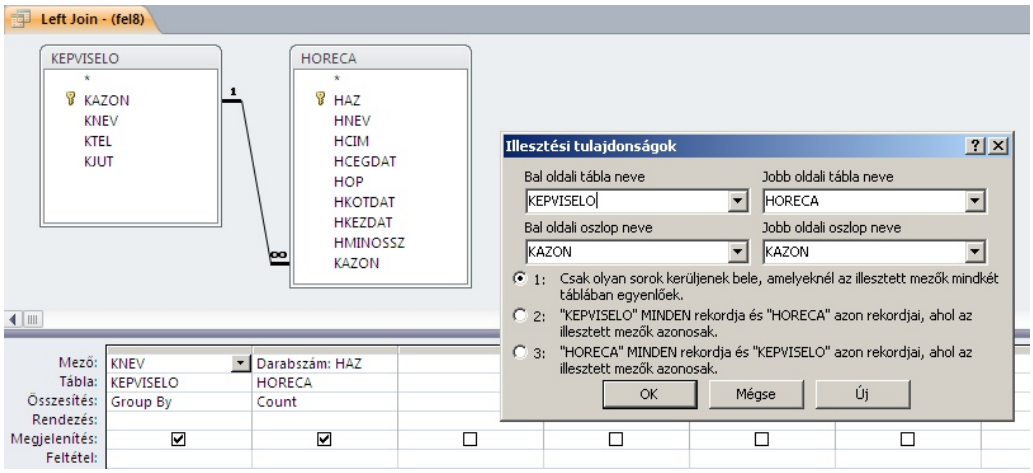
Fel kell vennünk mindkét táblánk. Szükségünk van a KEPVISELO.KNEV és a megkötött biztosítások összeszámlálásához a HORECA.HAZ mezőkre. Utóbbi helyett más kötelező tulajdonságtípust is választhatnánk a HORECA egyedből, de mindenképpen ebből a táblából. A HAZ mezőnél a *Count()* függvényt kell beállítani.



30. ábra: 8. feladat – inner join

Nyissuk meg a KEPVISELO táblát. Képviselőink között megtalálható Clarice Starling, a Bárányok hallgatnak híres ügynöknője. Viszont ő nem jelent meg a lekérdezésben, bár a feladat az összes képviselő nevét kérte és mi be is állítottuk. Miért nincs ott Clarice Starling? Nem kötött egyetlen szerződést sem. Ezt a KEPVISELO tábla rekordjainál lenyitható *segédadatlappal* (*segédtáblával*) könnyen ellenőrizhetjük. Ettől persze még a feladat igényelné a megjelenítést, a kapcsolat alapértelmezett beállítása azonban nem teszi ezt lehetővé. Ezt a kapcsolatok definiálásánál (24. ábra) az *Illesztés típusa*... gombra kattintva szabályozhatjuk. Ha azonban nem általános érvényűen szeretnénk változtatni, akkor helyileg, egy adott lekérdezésnél is felülírhatjuk a kérdéses értéket.

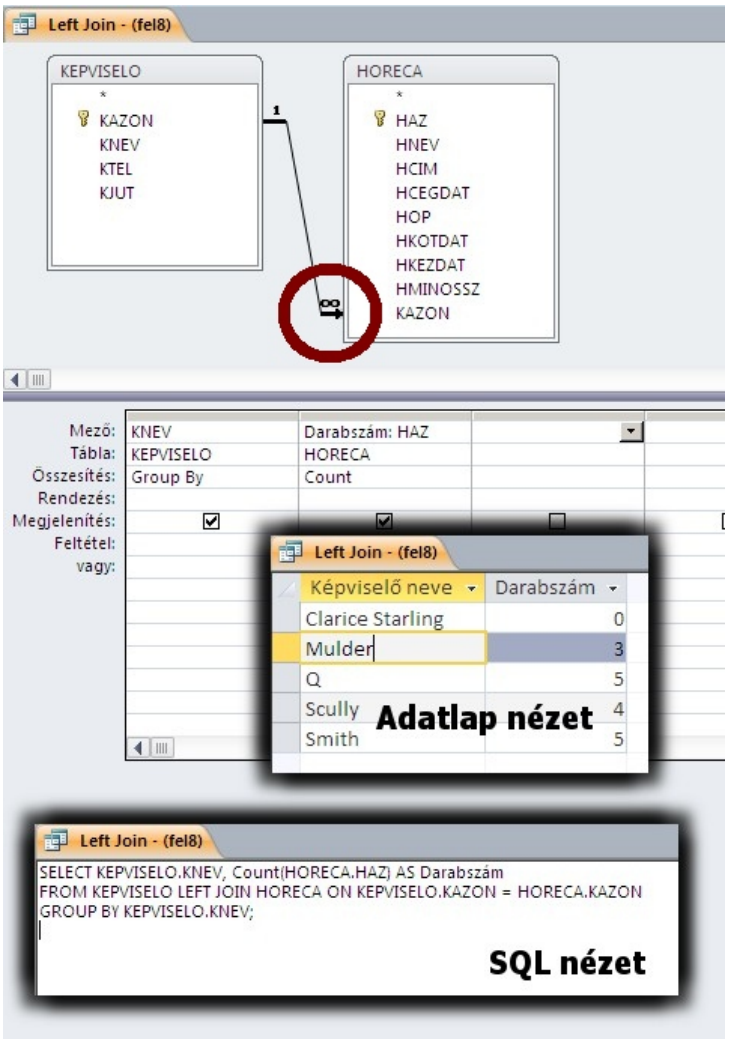
Kattintsunk a két táblát kapcsolatát szimbolizáló összekötő vonalra. A felugró ablak az alábbi ábrán látható:



31. ábra: 8. feladat – illesztési tulajdonságok

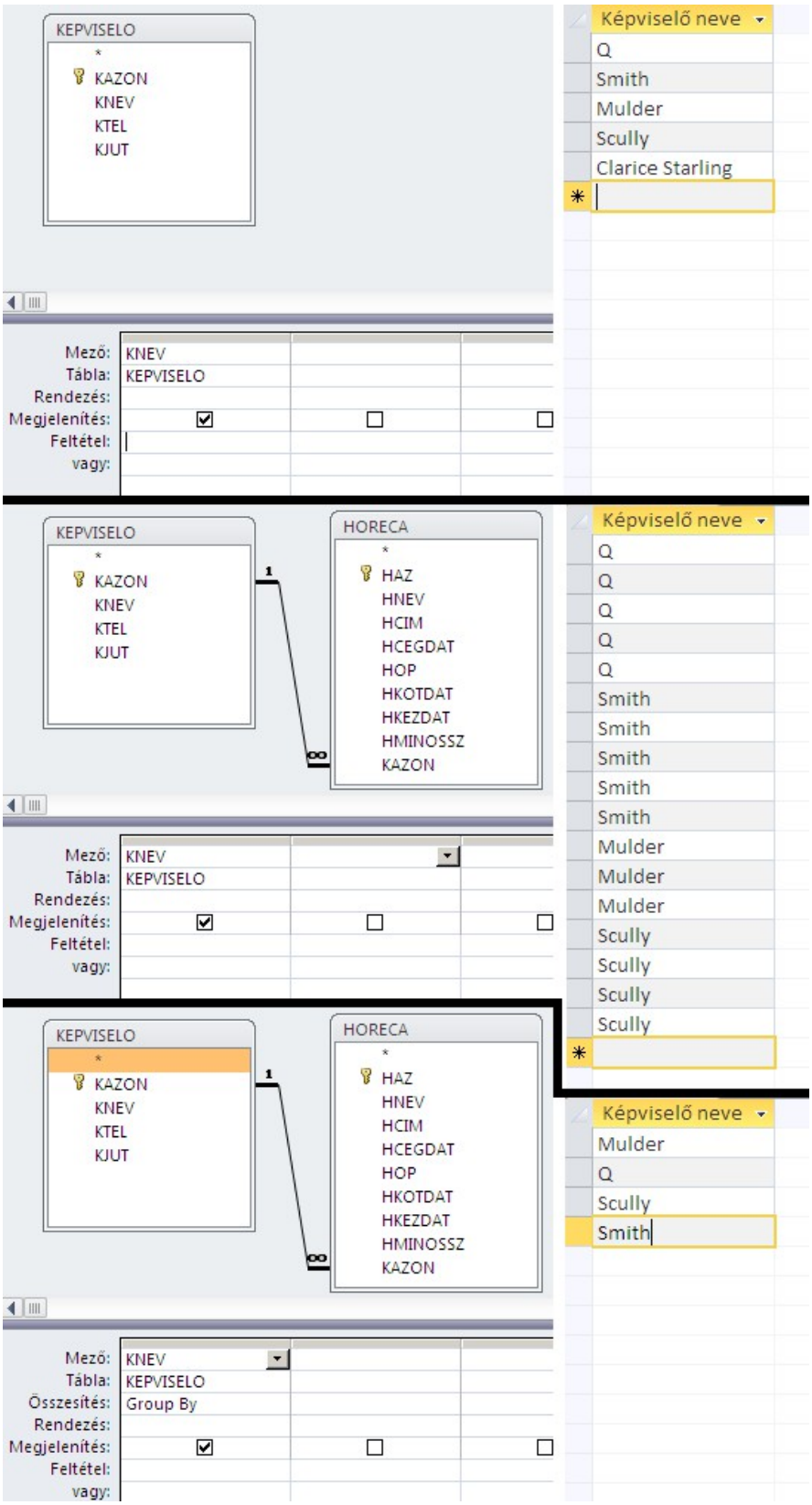
Alapesetben csak azon rekordok kerülnek a lekérdezésbe, melyek a kapcsolat mindkét oldalán értékkel rendelkeznek (inner join). A rendszer először összefűzi a rekordokat, majd a megadott feltételek alapján kiválogatja, összesíti őket. Azon sorok, amelyekhez nem kapcsolódik a másik tábla egyetlen eleme sem, már az összekapcsolásból kimaradnak. Jelen példában ez csak a KEPVISELO tábla adataira lehet érvényes, a hivatkozási integritás megkövetelése miatt a HORECA táblában nem létezhet kapcsolt képviselő nélküli szerződés.

Lehetőségünk nyílik azonban azt kérni, hogy a KEPVISELO tábla minden rekordja kerüljön be a lekérdezésbe (left join). A bal oldali tábla minden esetben az „egy”, a jobb oldali pedig a „több” oldala a kapcsolatnak, függetlenül a vizuális elrendezéstől, bár mi a fenti ábrán a vizuálisan is így rendeztük a táblákat. Ha tehát a második opciót választjuk, Clarice Starling is bekerül az eredmények közé nulla szerződés darabszámmal. Az SQL-ben az INNER szó LEFT-re cserélődik. Mivel a tervező és SQL nézet ugyanarra a lekérdezésre vonatkozik, ha az SQL-t írtuk volna át, akkor tervező nézet illesztési tulajdonságok ablaka értelemszerűen módosul, az eredmény ugyanaz. A kapcsolat változtatását az összekötő vonalon megjelenő nyílfej jelzi.



32. ábra: 8. feladat – left join

Ha definiáljuk a kapcsolatokat – mint ahogy a 2. feladatban láthattuk – akkor egy, a lekérdezésbe betett felesleges tábla nehezen észrevehető hibát okozhat. Ennek bemutatására egy egyszerű lekérdezés szolgál: kérdezzük le a képviselőink nevét. Az alábbi ábrán láthatjuk, hogy mi lesz az eredménye, ha a HORECA táblát is beletesszük a lekérdezésbe – holott nem szükséges. Az inner join kapcsolat miatt Clarice Starling már a két tábla rekordjainak összekapcsolásakor kiesik, nem kerül be a lekérdezés vizsgálatának kiinduló halmazába sem. Ezért a táblák hozzáadását is át kell gondolni, nem helyes mindig az összes táblát minden lekérdezéshez hozzáadni. Ha SQL kódokat írnánk, sem tennénk ezt.



33. ábra: 8. feladat – táblák illesztésének hatása a lekérdezésekre

A felső részen látható lekérdezés a megfelelő, a mellette látható adatsorral tér vissza. Ha bevonjuk a HORECA táblát is – bár a lekérdezésben nincs szerepe – az összekapcsolt rekordokban már többször jelenik meg egy-egy képviselő neve, ez látható az ábra középső részén. Itt már nem szerepel Clarice Starling az inner join kapcsolat korábban kifejtett hatása miatt. Ha SQL nézetben kézzel bevinnénk a

DISTINCT predikátumot a SELECT utasításba, akkor a rendszer elhagyná a többszörös rekordokat. A DISTINCT predikátum alkalmazásakor a SELECT utasításban szereplő mező(k) értékei (több mező esetén a kombinációjuknak) közül csak az egyediek jelennek meg. Viszont ez esetben is csak azon képviselők neve jelenik meg, amelyekhez legalább egy szerződés tartozik (tehát Clarice Starling nem).

```
SELECT DISTINCT KEPVISELO.KNEV
FROM KEPVISELO INNER JOIN HORECA ON KEPVISELO.KAZON = HORECA.KAZON;
```

Ha beállítjuk ez esetben a Group By összesítést – az ábra alsó része – úgy az képviselők neve csak egyszer kerül kiírásra, de Clarice Starling továbbra sem szerepel, hiszen a neve már a lekérdezés kiindulását jelentő összekapcsolt rekordhalmazba sem került bele. Ez az eredmény megegyezik a DISTINCT predikátum alkalmazásával. Előre definiált kapcsolatok esetén a feleslegesen hozzáadott táblák tehát megváltoztathatják az eredményt, amit éles rendszerben nehéz észrevenni. Nem véletlenül kezdtük az előző feladatokat a szükséges táblák körének pontos meghatározásával.

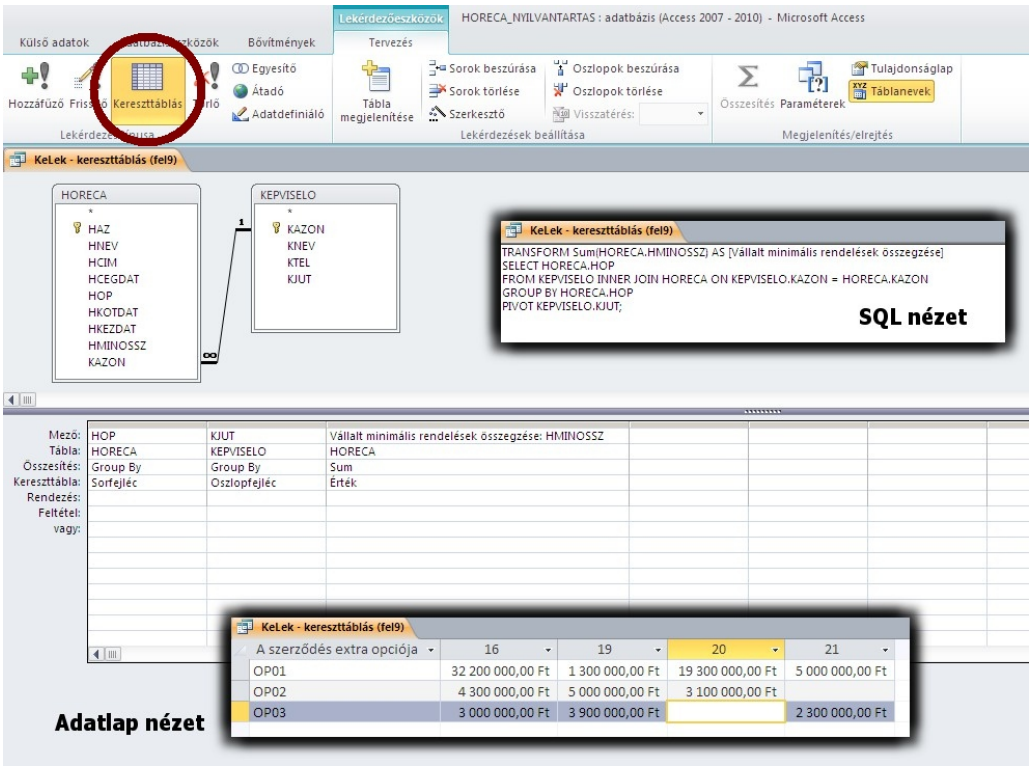
9. feladat – kereszttáblás lekérdezés

Készítsen KeLek néven kereszttáblás lekérdezést, amely a választott extra opció és a képviselő jutalékának függvényében kiírja a vállalt minimális rendelések összesítését!

A KJUT, HOP és HMINOSSZ mezőkre van szükségünk, azaz mindkét táblát hozzá kell tennünk a lekérdezéshez. Hozzuk létre a lekérdezést, majd a *Lekérdezőeszközök/Tervezés* lapon válasszuk a *Kereszttáblás* opciót. Első ránézésre a tervezőrács adatai változtak.

A kereszttáblás lekérdezés az adatokból kimutatás jellegű nézetet állít össze. Ennek megfelelően lehetőségünk van sor-, valamint oszlopfejlécek megadására. Ide olyan csoportosító tulajdonságokat érdemes felvenni, amelyek relatíve kevés különböző adatot tartalmaznak sokszor. A választott extra opció mezője (HOP) például ilyen, általa lehetőség van a vizsgált összesítés megbontására. A képviselő jutaléka már kevésbé jó, de mivel a kereszttáblához legalább két csoportosító mezőre van szükség, a feladat ezt is kéri. Ha például csak néhány különböző jutalékkulcsot alkalmaznánk sok-sok képviselőnknél, akkor tökéletes csoportosító mező lenne. Jelen esetben viszont kevés számú képviselőnk van és mindegyiknek más jutaléka, ezért itt nem tökéletes ez a választás. A sor- és oszlopfejlécek megadása önkényes, a tábla elrendezését változtatja, ha ezeket a választott mezőkön belül felcseréljük. Akár több sor- és oszlopfejléc is lehet, a kereszttábla dimenziószáma – a csoportosító tulajdonságtípusok száma – lehet kettőnél több.

A tervezőrácsra vitt harmadik oszlopunk a HMINOSSZ összegzése lesz, ez tehát Sum összesítéssel, a kereszttábla *értékhelyére* kerül. Bár egy bevitt *érték* típus esetén nem jelenik meg, megadtuk a mező egyéni elnevezését is. *Érték* típusú kereszttábla-mezőből is lehet egynél több. A választó lekérdezésekhez hasonlóan itt is lehet feltételekkel, vagy akár paraméterekkel élni, azonban jelen feladat ilyen kitételt nem tartalmaz.



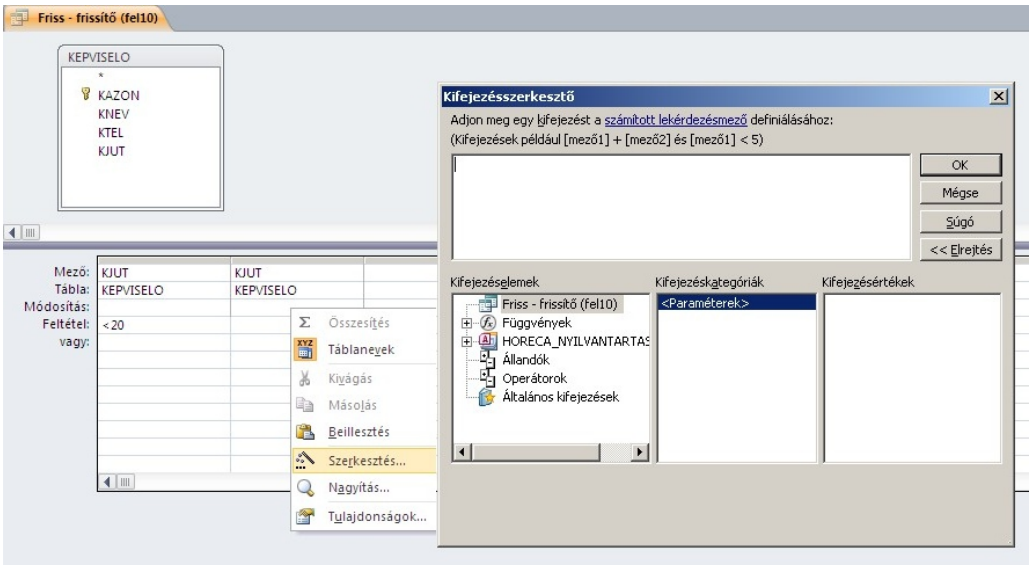
34. ábra: 9. feladat – kereszttáblás lekérdezés

10. feladat – frissítő lekérdezés

Készítsen Friss néven lekérdezést, amely 2 ezrelékkel növeli a 20 ezrelék alatti jutalékkal rendelkező képviselők jutalékát!

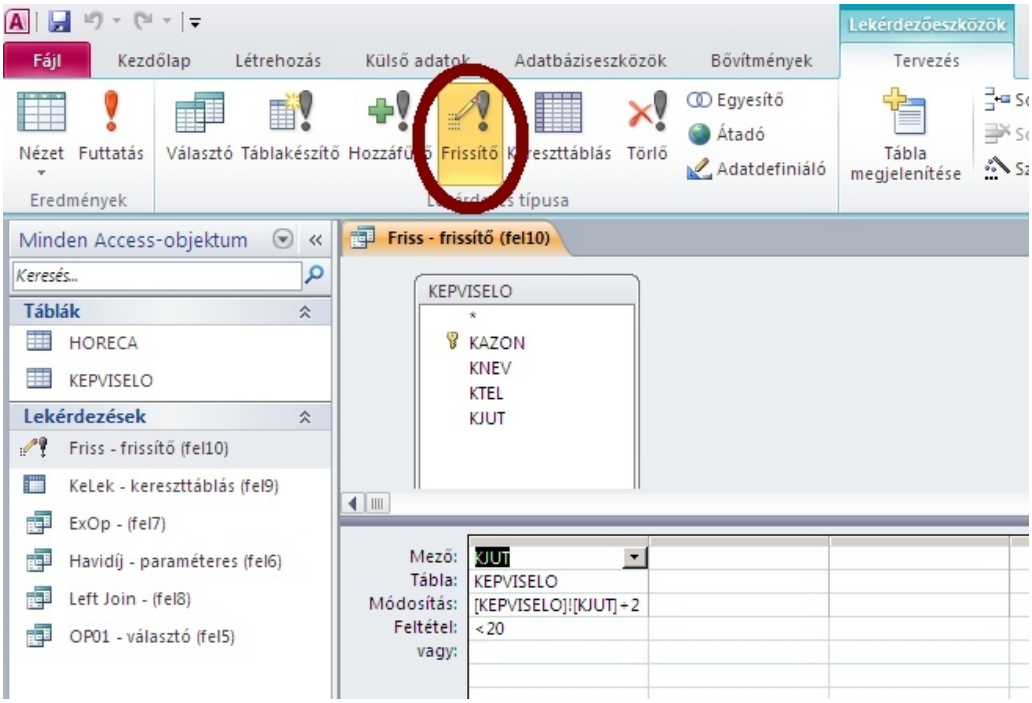
A lekérdezéseket leggyakrabban adatok lekérésére használjuk, ám lehetőség nyílik a táblaszerkezet, illetve az adattartalom lekérdezésből történő módosítására is. Frissítő lekérdezésnél tulajdonképpen két műveletet kötünk össze: bizonyos feltételeknek megfelelő rekordokat kérünk le, majd – mivel nem a listára vagyunk kíváncsiak, hanem ezek adataiban tennénk módosítást – azonnal változtatunk előre definiált szabályok mentén.

Hozzunk létre egy új lekérdezést a KEPVISELO táblával, mivel csak a jutalék mezőre (KJUT) lesz szükségünk! Válasszuk a *Lekérdezőeszközök/Tervezés* lapon a *Frissítő* gombot. A tervezőrácson egy *Módosítás* opciót kaptunk, a *Megjelenítés* lehetősége pedig eltűnt. A feltétel beállításához fel kell vennünk a KJUT mezőt, majd a feltételhez be kell állítanunk, hogy <20. A módosítás meghatározásához újfent vegyük fel a KJUT mezőt. A módosítás képletéhez jó lenne egy segéd. A bevezető feladatsornál említésre került, hogy a jobb egérkattintással előhívható helyi menüből előhívhatjuk a már megismert kifejezésszerkesztőt. (Ezt a korábbi feladatoknál is megtehettük volna.)



35. ábra: 10. feladat – a kifejezésszerkesztő lekérdezéseknél

Az összeállítandó kifejezés: *[KEPVISELO]![KJUT] + 2* . A szükséges mezőt a HORECA_NYILVANTARTAS.accdb/Táblák/KEPVISELO/KJUT úton tudjuk behívni.



36. ábra: 10. feladat – frissítő lekérdezés

Az adatlap nézet itt másodlagos jelentőségű. Tulajdonképpen a frissítő lekérdezésbe bekerülő adatokat nézhetjük meg úgy, mintha választó lekérdezésben lennénk. A lekérdezés mentése és bezárása után futtathatjuk a lekérdezést. A figyelmeztető üzenetek is jelzik, amit korábban írtunk. Itt éles rendszerben dolgozunk, ahol nincs lehetőség visszavonásra. Összesen két 20 ezrelék alatti adat frissül, az egyik ezzel 20 fölé kerül. Így az esetleges második futtatáskor már csak egy rekord frissülne, azt követően pedig már egy sem. A későbbi adategyezésekhez szükséges információ: mi összesen egyszer frissítettünk e lekérdezéssel.

Frissítő lekérdezésnél is lehetőség van paraméterek futásidőben történő bekérésére. Pl. megpróbálkozhatunk a feladatban írt 20, vagy 2 értékek paraméterekkel történő helyettesítésével.

Nézzük meg még az SQL utasítást:

```
UPDATE KEPVISELO SET KEPVISELO.KJUT = [KEPVISELO]![KJUT]+2
WHERE (((KEPVISELO.KJUT)<20));
```

A frissítést az UPDATE .. SET .. utasítás jelenti, a WHERE záradék feltételeivel korlátozhatjuk az érvényességi tartományt.

11. feladat – törlő lekérdezés

Készítsen TorL néven lekérdezést, amely törli az 1997.01.01 előtt bejegyzett vállalkozások szerződéseinek adatait a HORECA táblából!

Az űrlapok az egyes rekordok áttekinthető megjelenítésére szolgálnak, a jelentések pedig a lekérdezések adatait szervezik strukturált formába. Ha az adatbázist valamilyen feltételrendszer mellett módosítanánk, arra az űrlap kevéssé alkalmas, ott soronként kellene a változtatásokat megejteni. A lekérdezések segítségével meghatározott feltételrendszer mentén állítunk össze nézeteket. Logikus elvárás, hogy az így lekért adatok együttes frissítésére, törlésére is lehetőség nyíljon. Az SQL és az Access is lehetőséget nyújt ezekre. E funkciókat például egy-egy nyomógombhoz, vagy menühöz rögzítve tehetjük elérhetővé a felhasználók részére. Jelen feladat a törlő lekérdezést mutatja be. A feltétel úgy került kialakításra, hogy csak egyetlen rekord törlődjön, kellő rekordunk maradjon a későbbi tesztelésekre.

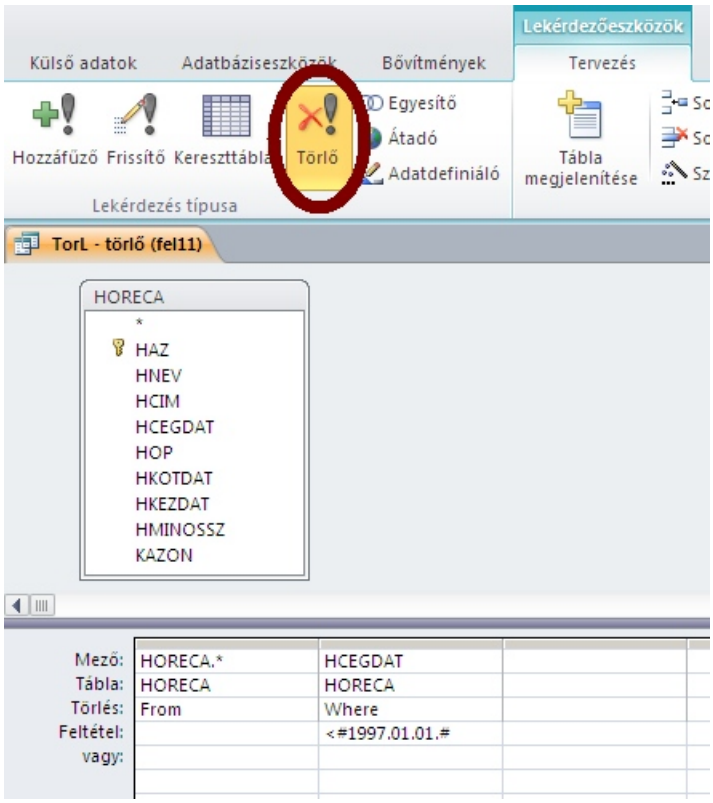
Hozzunk létre egy új lekérdezést a HORECA táblával, mivel csak a cégbejegyzés mezőre (HCEGDAT) lesz szükségünk. Válasszuk a *Lekérdezőeszközök/Tervezés* lapon a *Törlő* gombot. A tervezőrácson egy *Törlés* opciót kaptunk, a *Megjelenítés* lehetősége pedig eltűnt. A *Törlés* legördülő menüjében a WHERE és a FROM opciók közül választhatunk. Milyen feltételek mentén, mit töröljön.

WHERE – feltétel: A HCEGDAT mező <#1997.01.01.#

FROM – mit, honnan: A teljes rekordot, tehát a HORECA.*-ot kell törölni.

Az adatlap nézet itt is csak betekintésre szolgál: futtatás esetén mely rekordok kerülnek törlésre. Ellenőrzésképp érdemes megnézni, hogy valóban csak egy szerződés, az AA cégé törlődik.

Itt is lehetőség van paraméterek futásidőben történő bekérésére. Megpróbálkozhatunk a feladat feltételének paraméteres helyettesítésével, de ügyeljünk arra, hogy a változtatások az éles adatbázist érintik.



37. ábra: 11. feladat – törlő lekérdezés

Zárjuk be, majd futtassuk a lekérdezést. Egy sor kerül törlésre.

A megfelelő SQL parancs:

```
DELETE HORECA.*, HORECA.HCEGDAT
FROM HORECA
WHERE (((HORECA.HCEGDAT)<#1/1/1997#));
```

A felesleges zárójelek – csakúgy, mint korábban – elhagyhatók. A DELETE utasításban elegendő lenne a DELETE HORECA.* . A HCEGDAT a * része, felesleges külön kiemelni (csak az Access ragaszkodik hozzá, de ha kézzel illesztenénk be az SQL utasítást, működne, maximum a rendszer kiegészítené a felesleges elemekkel).

12. feladat – hozzáfűző lekérdezés

Másolja le a HORECA táblát, majd készítsen HozFuz néven lekérdezést, amely a másolat adatait hozzáfűzi az eredeti HORECA táblához!

A másolás a bal oldali navigációs sávon a HORECA táblára állva a CTRL+C, CTRL+V billentyűparancsokkal elvégezhető. A tábla struktúráját és az adatokat is másoljuk, a tábla neve legyen xH_MASOLAT. (A tábla így az ábécé végére kerül.)

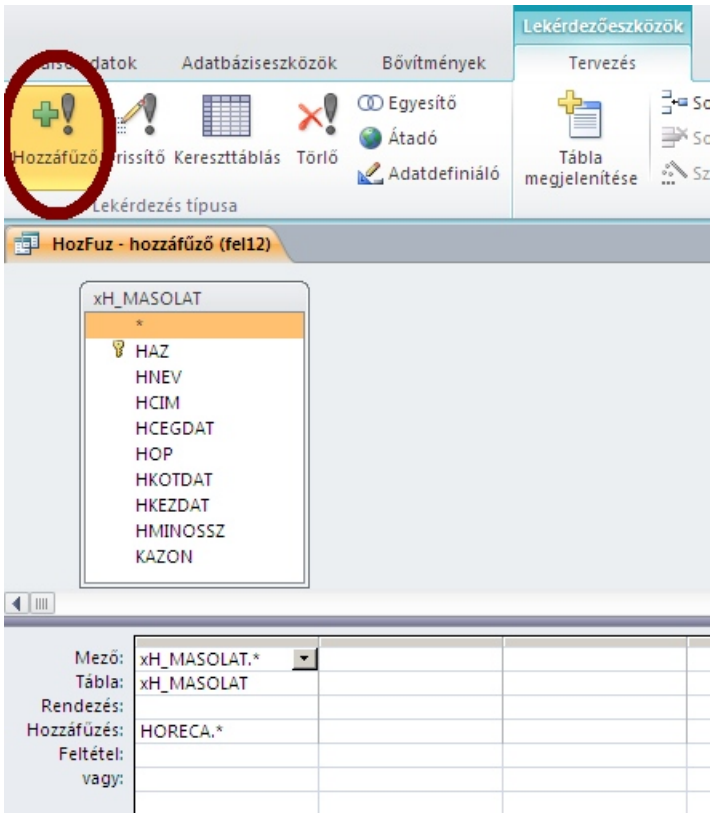
Hozzunk létre egy új lekérdezést az xH_MASOLAT táblával. Válasszuk a *Lekérdezőeszközök/Tervezés* lapon a *Hozzáfűző* gombot! A felugró, hozzáfűzés párbeszédpanelen adhatjuk meg, hogy az adott lekérdezés mely tábla rekordjaihoz csatoljon új rekordokat. Itt a HORECA táblát kell megadnunk, ehhez szeretnénk a lemásolt adatokat hozzáfűzni.

A tervezőrácson egy *Hozzáfűzés* opciót kaptunk, a *Megjelenítés* lehetősége pedig eltűnt. A lekérdezésünk végtelenül egyszerű. Az xH_MASOLAT.*-ot szeretnénk a HORECA táblához hozzáfűzni, a teljes rekordhalmazt, így a hozzáfűzésnél a HORECA.*-ot választjuk.

Feltételeket fogalmazhatnánk meg a hozzáfűzendő rekordok szűkítésére, akár paraméteres megoldással is.

Az adatlap nézet itt is csak a hozzáfűzendő rekordok megtekintésére szolgál.

Mi fog történni, ha futtatjuk a lekérdezést?



38. ábra: 12. feladat – hozzáfűző lekérdezés

Zárjuk be, majd futtassuk a lekérdezést! Mivel a lemásolt adattáblában található rekordok elsődleges kulcsai megegyeznek az eredeti tábla elsődleges kulcsaival, ezért egyetlen sor sem kerül beillesztésre. Ez a feladat a hozzáfűző lekérdezés bemutatására szolgált, valós hozzáfűzést csak az elsődleges kulcsok módosításával tudnánk megvalósítani – ha más szabályt nem szegünk meg, pl. más mezőben nincsen beállítva egyedi (nem lehet azonos) index.

Az SQL utasítás:

```
INSERT INTO HORECA
SELECT xH_MASOLAT.*
FROM xH_MASOLAT;
```

A SELECT utasításban megvalósított lekérdezést szúrjuk be (INSERT INTO) a megadott táblába, mezőkbe. A SELECT utasítás, mint választó lekérdezés az ott megismert teljes „eszköztárat” felvonultathatja.

13. feladat – táblakészítő lekérdezés

Hozzon létre egy TabKesz nevű lekérdezéssel egy új LEKERDEZETT nevű táblát, amely a szerződések azonosítóját, a szerződő cég nevét, bejegyzésének dátumát, valamint a szerződést kötő képviselő nevét és telefonszámát tartalmazza!

Hozzunk létre egy új lekérdezést a HORECA és KEPVISELO táblákkal, a feladatban kért mezők mindkét táblát érintik. Válasszuk a *Lekérdezőeszközök/Tervezés* lapon a *Táblakészítő* gombot. A

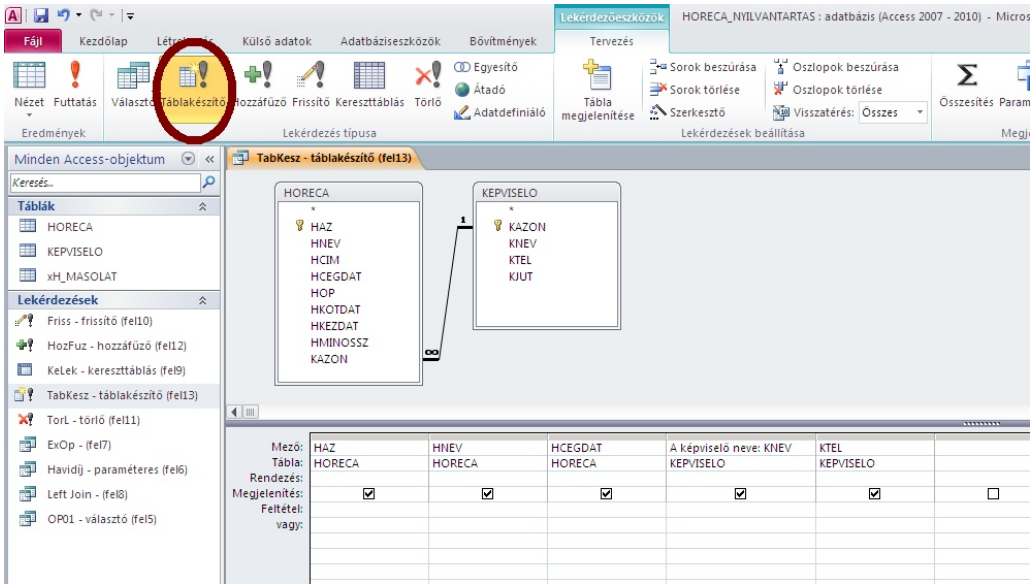
felugró, táblakészítő párbeszédpanelen adjuk meg a LEKERDEZETT nevet! A legördülő menü ellenére gépelnünk kell.

A feladatban kért mezők: HORECA.HAZ, HORECA.HNEV, HORECA.HCEGDAT, KEPVISELO.KNEV és KEPVISELO.KTEL. Állítsuk be ezeket a tervezőrácson. Az adatlap nézeten megnézhetjük a készítendő tábla adattartalmát.

A táblakészítő lekérdezés tulajdonképpen olyan választó lekérdezés, amely az eredményül adott nézetet fizikailag is az adatbázisba menti. Minden olyan lehetőséggel élhetnénk, amelyet a választó lekérdezésnél megismertünk. Feltételeket fogalmazhatnánk meg, akár futtatáskor bekért paraméterek segítségével is. Lehetőségünk van összesítésre, számításokra. Az eredménytábla rendezhető, továbbá a tervezőrácsra vitt elemeket nem kötelező megjeleníteni a készítendő új táblában.

Ennek megfelelően az SQL utasítás csak annyival bővül, hogy a SELECT utasítás végén megadjuk (INTO) a készítendő tábla nevét:

```
SELECT HORECA.HAZ, HORECA.HNEV, HORECA.HCEGDAT,
KEPVISELO.KNEV AS [A képviselő neve],
KEPVISELO.KTEL INTO LEKERDEZETT
FROM KEPVISELO INNER JOIN HORECA ON KEPVISELO.KAZON = HORECA.KAZON;
```



39. ábra: 13. feladat – táblakészítő lekérdezés

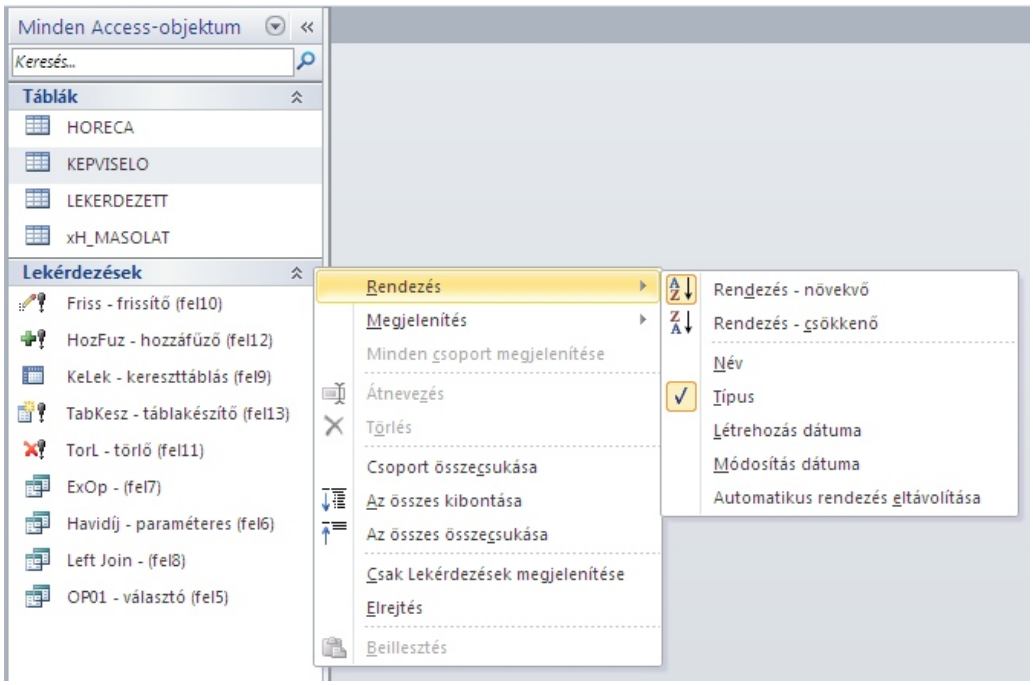
Zárjuk be, majd futtassuk a lekérdezést, ellenőrizzük a létrehozott új táblát!

A lekérdezés a mezőtulajdonságok jelentős részét nem vitte át az új táblába, ezeket újra definiálnunk kellene. A mezőcímek sem kerültek át, bár adatlap nézetben még ezek voltak az oszlopfejlécek, a létrehozott új táblába csak a (nagybetűs) mezőnevek kerültek át. Egyedül a KNEV mezőnél van másmilyen név, ott ez a lekérdezésben definiálva lett. Viszont ha megnézzük az új tábla tervező nézetét, láthatjuk, hogy nem a KNEV cím mezőtulajdonságába került bele a név, hanem a KNEV mezőnév került felülírásra.

Ha a megadott táblanév létezik, a lekérdezés futtatása törli a régi táblát (természetesen csak egy figyelmeztető üzenet jóváhagyása után).

A navigálás panel rendezése

Az elkészített lekérdezések sorrendje látszólag nem logikus. A típus szerinti – Frissítő, Hozzáfűző, Kereszttáblás, Táblakészítő, Törlő, Választó – ábécé rendezést a helyi menüben változtathatjuk.



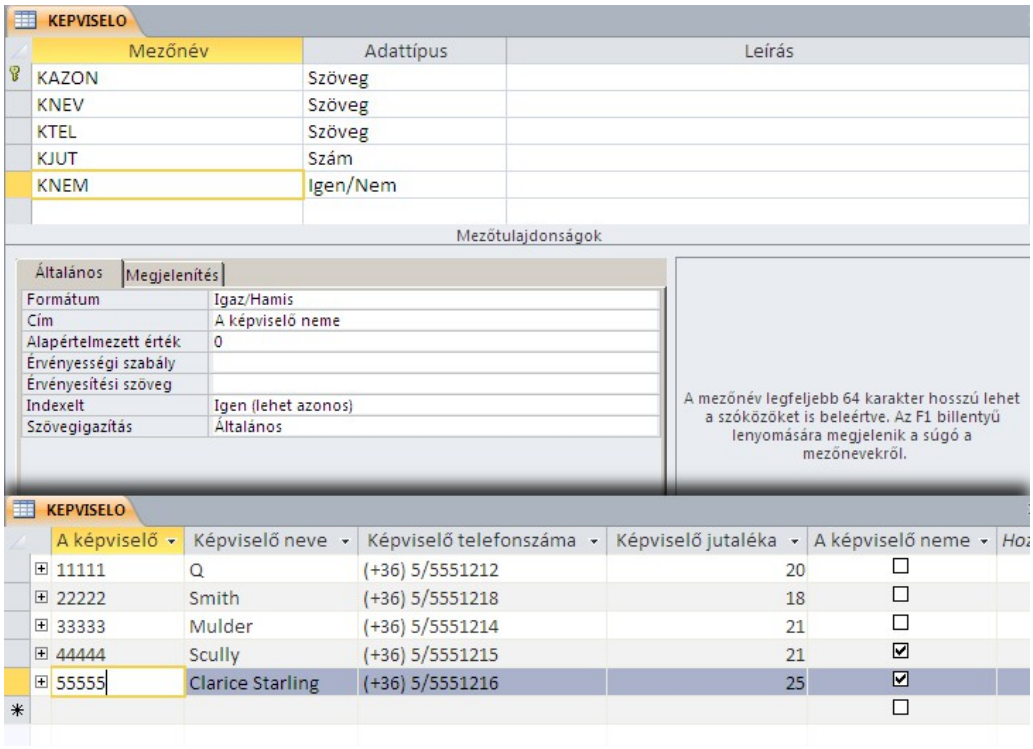
40. ábra: Access objektumok rendezése

14. feladat – űrlap készítése 1.

A KEPVISELO táblához vegyen fel egy új tulajdonságtípust, melyben a képviselő neme szerepel (KNEM – Igen/Nem adattípussal)!

Készítsen Urlap_Segeddel néven űrlapot segédűrlappal a KEPVISELO és HORECA tábla adataiból (egy adat csak egy helyen szerepeljen)! Tegyen egy vezérlőelemet az űrlapra, mely kiírja, hogy a képviselő „férf”’ vagy „nő”! E vezérlőelem ne kapjon fókuszt! Állítsa a bejárási sorrendben, hogy a csatolt űrlapra kerüljön legvégül a fókusz, illetve, hogy a bejárás végén ne a következő rekord jöjjön! Állítsa be, hogy a 20-nál (ezrelékben) nagyobb jutalékkal rendelkező képviselők jutaléka vastag piros számokkal jelenjen meg (feltételes formázással)!

Nyissuk meg a KEPVISELO táblát tervező nézetben és vegyük fel a kért mezőt!

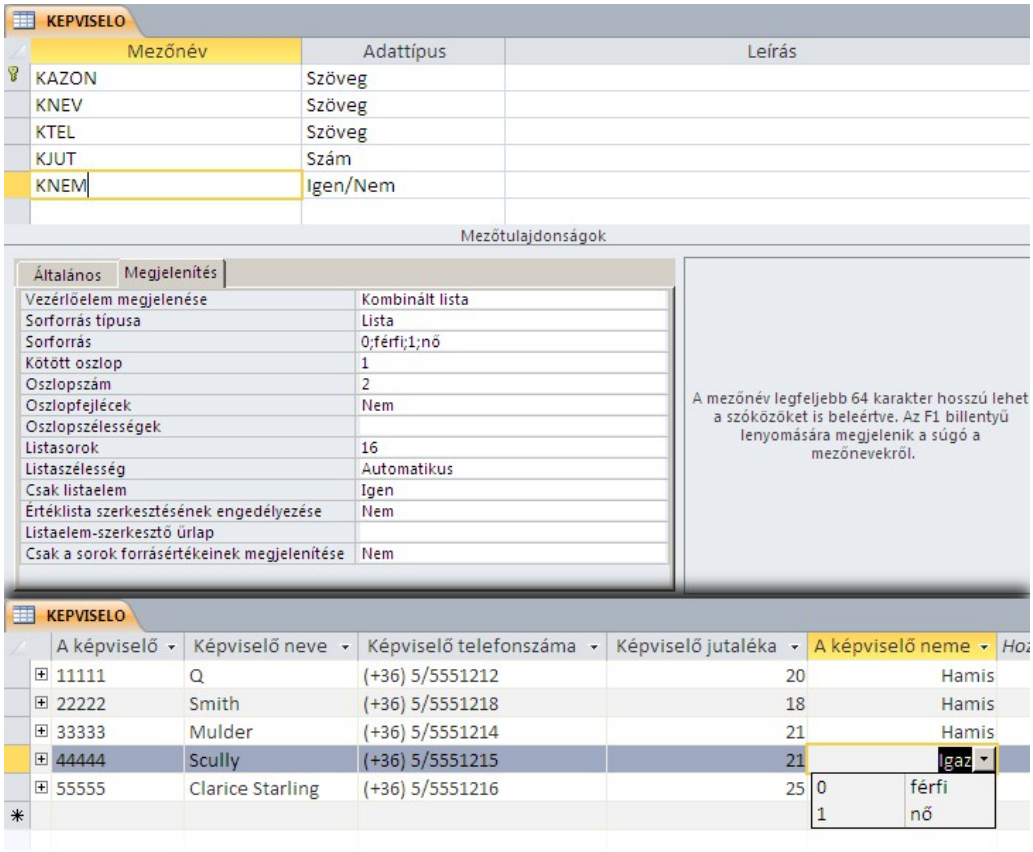


41. ábra: 14. feladat – az új tulajdonságtípus hozzáadása

Mentsünk, lépjünk át adatlap nézetbe és állítsuk be a nemeket! Az üres négyzet 0-nak, a „pipa” 1-nek (-1-nek) felel meg. A mező címébe írhattuk volna, hogy „A képviselő hölgy?”, ez esetben egyértelmű lenne mit jelent az igen. Nem így jártunk el, ez kevésbé szerencsés, így figyelnünk kell a későbbiekben. Megállapodás kérdése, melyik nemet hogyan jelöljük. Legyen a „pipa” a hölgy. Jelöljük be a női képviselők (Mulder, Clarice Starling) jelölőnégyzetét.

Ha tervező nézetben a mezőtulajdonságok megjelenítés fülén kombinált listát állítunk be vezérlőelemnek, a bevitelt csak a listaelemekre korlátozzuk, a sorforrás típusát pedig listára állítjuk, akkor lehetőségünk nyílik egyéni felvitel megvalósítására. Az alábbi ábrán két oszlopot állítottunk be, ahol az első a kötött oszlop, azaz az adott a legördülő menü (kombinált lista) adott sorának kiválasztásakor az ebben szereplő érték kerül a cellába. Az ábra alsó részén látható adatlap nézetben figyelhetjük meg az eredményt. A táblázatos bevitelkor a szöveges megnevezést is látjuk, viszont a cellába a számértékek kerülnek be. Kombinált lista esetén az *Igaz*, illetve *Hamis* értékek látszanak a jelölőnégyzetek helyett.

Újfent kiemeljük, hogy a felhasználók űrlapokon és jelentéseken keresztül érik el az adatbázist, jelen megoldást inkább csak a lehetőségek szemléltetése kedvéért valósítottuk meg. Ugyanez indokolja az *Igen/Nem* adattípus használatát is.



42. ábra: 14. feladat – a KNEM tulajdonságtípus megjelenítési lehetőségei

(Kis kitérő: A KNEM mező összesen kétféle egymástól különböző adatot tartalmazhat, ezeket viszont sokszor, tehát tökéletes csoportosító mező, feltárja az adathalmaz nem szerinti struktúráját. A kereszttáblás lekérdezésnél említettük, hogy az ott használt képviselő jutaléka (KJUT) erre kevésbé alkalmas. Gyakorlásként készíthetünk egy kereszttáblás lekérdezést KeLeK2 néven a vállalt minimális rendelések összesítéséről a választott extra opció mezője (HOP) és a képviselő neme alapján. Értelmezzük is a kapott eredményeket!)

A feladat hátralévő részeit az alábbi videón mutatjuk be:

15. feladat – űrlapok nézetei

Tiltsa le a fenti űrlap adatlap, kimutatás és kimutatásdiagram nézetét!

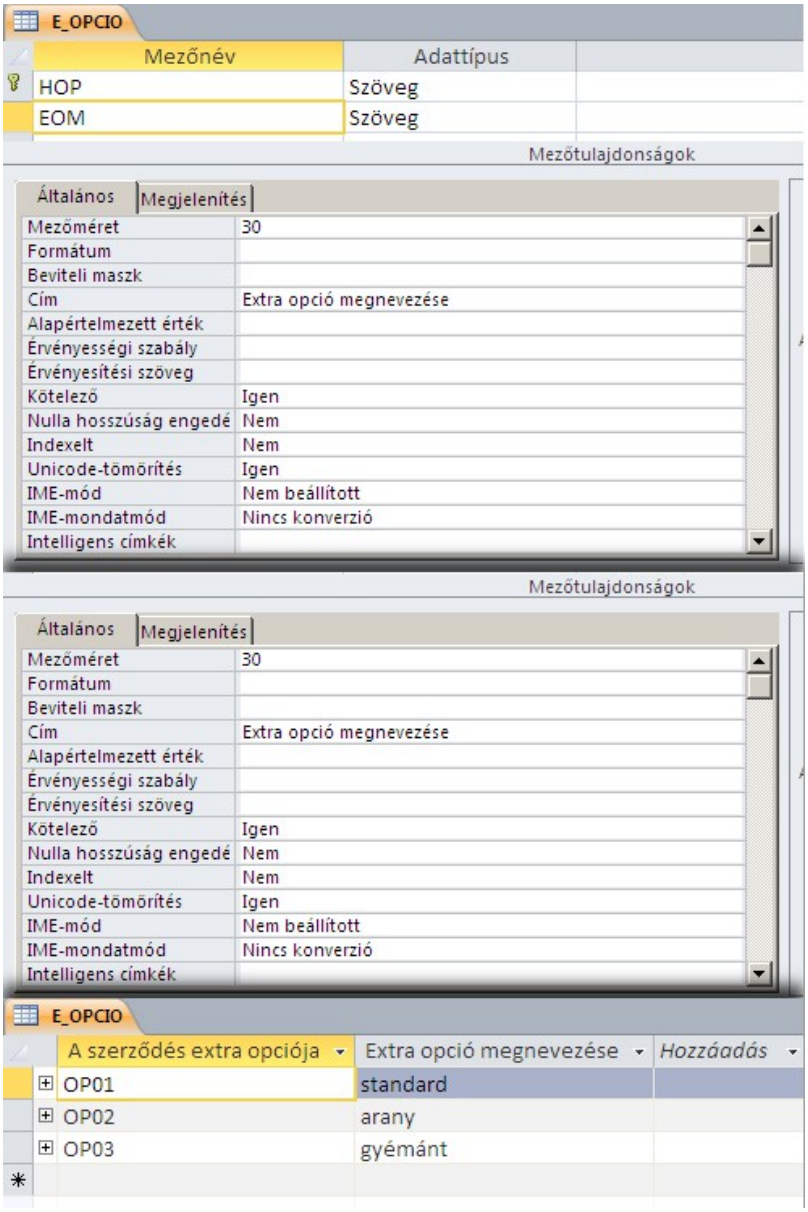
A feladatot az alábbi videó foglalja össze:

16. feladat – űrlap készítése 2.

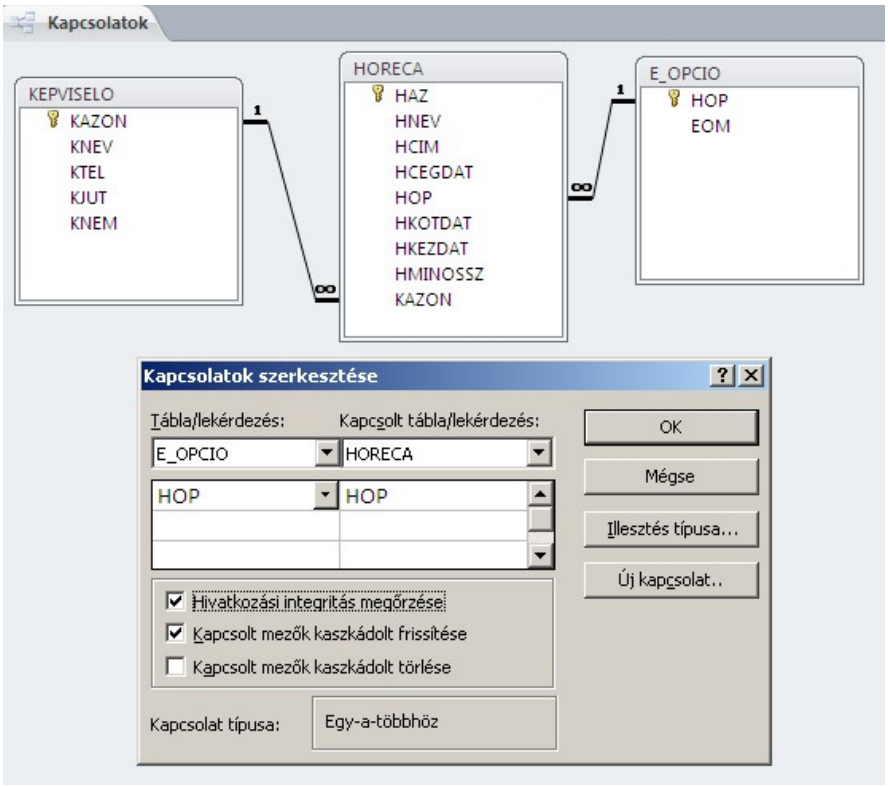
Készítsen E_OPCIO néven új táblát, melybe az extra opció kódja (HOP – lásd a HORECA táblában), és az opció megnevezése kerül (standard, arany, gyémánt a kód növekvő sorrendjében). Hozza létre a kapcsolatot, állítsa be erre a kapcsolatra is a hivatkozási integritást!

Tervező módban hozzon létre TERVEZETT nevű űrlapot a HORECA tábla adatainak kezelésére! Az űrlap rendelkezzen űrlapfejjel! Készítsen legördülő menüs kombinált lista vezérlést úgy, hogy a kezelő csak az extra opció megnevezését lássa ! Az űrlap legyen modális és előugró!

A feladat első része ismétlésnek tekinthető. A beállításokat az alábbi két ábra részletezi:



43. ábra: 16. feladat – az E_OPCIO tábla tervező és adatlap nézete



44. ábra: 16. feladat – az E_OPCIO tábla kapcsolata

A feladat hátralévő részeit az alábbi videón mutatjuk be:

Megtehettük volna, hogy nem szerepeltetünk minden mezőt az űrlapon. Például a szerződés azonosítóját a rendszer adja. Ha nem szeretnénk ennél változtatást engedélyezni, annak a legjobb módja, ha nem is tesszük az űrlapon hozzáférhetővé. Vagy úgy tiltjuk le, mint a képviselő azonosítóját és nevét, vagy egyszerűen meg sem jelenítjük, nem kerül az űrlapra HAZ mező vagy kifejezés tulajdonságértékkel egyetlen vezérlő sem.

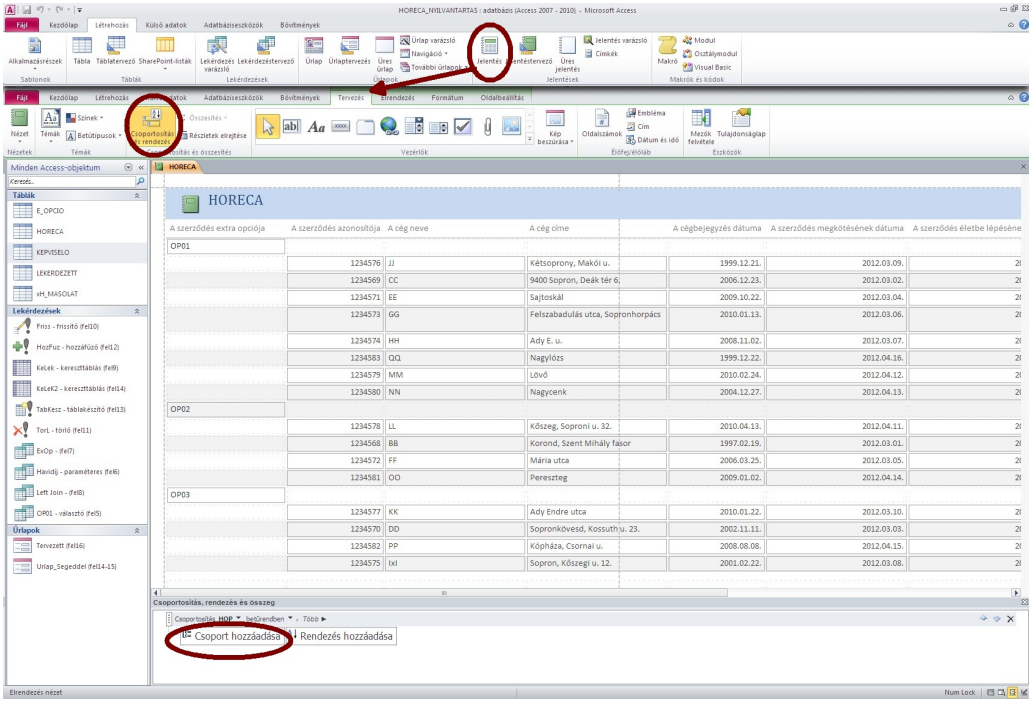
17. feladat – jelentés készítése

Készítsen jelentést ReszletesJelentes néven, mely tartalmaz jelentésfejet/lábat, oldalfejet/lábat, és képviselőnként, illetve extra opciónként tartalmazza az egyes biztosítási összegeket és a képviselő jutalékát! Képviselőnként írja ki az egyes képviselők összesített jutalékát is (Ft-ban)! A vállalt minimális rendelési összegeket külön oszlopban írja ki csoportonkénti összegzéssel, illetve folyamatos összegzéssel is!

Az adatok felvitelére, módosítására, törlésére az adott teendőnek megfelelő összeállítású űrlapokat készítünk a felhasználói munkákhoz. Ezen felül az adatbázisokból különböző struktúrájú információk kinyerésére is szükségünk lehet. Az ilyen igények formázott, rendezett és igény szerint összegzett adathalmazokkal történő kielégítésére szolgálnak a jelentések.

Hogy jobban lássuk, miről is van szó, e bekezdés erejéig ne foglalkozzunk a fenti feladattal. Álljunk a HORECA táblára a bal oldali navigációs panelen, majd válasszuk a *Létrehozás/Jelentés* gombot! El is készült az első jelentésünk, mely a HORECA-szerződések adatai összegzi. Most válasszuk az elrendezési nézetben megnyitott jelentés esetén elérhető *Jelentés-elrendezési eszközök/Tervezés/Csoportosítás és rendezés* gombot, majd az alul felnyíló részen a *Csoport beszúrása* opciót és ezután az előbukkanó mezőlistából a HOP mezőt. A jelentésünk az extra opció kódja szerint rendezett lesz. A képviselő kódja helyett például jobb lenne a képviselő nevét szerepeltetni, az extra opció kódja helyett pedig az előző feladatban felvett megnevezéseket. Ehhez a HORECA tábla mellett további táblák adatait is be kellene vonnunk a jelentés forrásába. Számos más változtatási igényünk

is lehetne, a fenti feladat megoldásán keresztül bemutatjuk az ezek kielégítésére szolgáló főbb lehetőségeket. Az összeállított első jelentésünket az alábbi ábra mutatja.



45. ábra: 17. feladat – az elkészült mintajelentés

Ezután kezdjünk neki a feladatnak. Válasszuk a *Létrehozás/Jelentéstervező* gombot! Egy üres jelentést kapunk tervező nézetben. Hívjuk be az *Jelentéstervező eszközök/Tervezés/Tulajdonságlap*-ot és a jelentés tulajdonságok adat fülén található rekordforrás tulajdonság mellett kattintsunk a …-ra. (A három pont bármely menüfelirat végén azt jelenti, hogy új ablak fog megnyílni.) A megnyíló lekérdezéstervezőben adjuk a lekérdezéshez a KEPVISELO, HORECA és E_OPCIO táblákat, a tervezőrács mező opciójába adjuk hozzá mindhárom táblából a „csillagot (*)” – minden mezőt. Az előző feladatban elkészített űrlap forrása ugyanez volt. Elkészíthettük volna e lekérdezést a többi lekérdezésünk közé mentve is, ez esetben elég lenne az erre épülő objektumok rekordforrásánál ugyanarra a lekérdezésre hivatkozni. Beágyazott formában a többi objektum elől rejtve marad. A rekordforrás tulajdonság tartalma:

```
SELECT HORECA.*, KEPVISELO.*, E_OPCIO.*
FROM KEPVISELO
INNER JOIN (E_OPCIO INNER JOIN HORECA ON E_OPCIO.HOP = HORECA.HOP) ON
KEPVISELO.KAZON = HORECA.KAZON;
```

A jelenleg még üres jelentésen a jobb egérgomb-kattintásra előugró helyi menüben válasszuk a Jelentésfej/-láb opciót! A feladatnak megfelelően ezt is be kell kapcsolnunk. Ezután az egyes mezőket, címkéket kell felvennünk. E tekintetben az előző feladatban tervező nézetben elkészített űrlap és a jelen feladatban tervező nézetben készítendő jelentés létrehozása között sok a hasonlóság. Vezérlőket a *Jelentéstervező eszközök/Tervezés/Vezérlők* csoport gombjaival tudunk felvenni. Vagy innen veszünk fel (kötetlen) beviteli mezőket, melyek mező vagy kifejezés tulajdonságát aztán a fent beállított rekordforrás lekérdezésből származó, a feladatban kért mezőkre állítjuk. Vagy a *Jelentéstervező eszközök/Tervezés/Mezők felvétele* gomb bekapcsolása után a mezőlistáról húzzuk a megfelelő felületre. A címkéknél használhatjuk a vezérlőelemek címkéjét, illetve a vezérlők csoportból is adhatunk hozzá címke-vezérlőt.

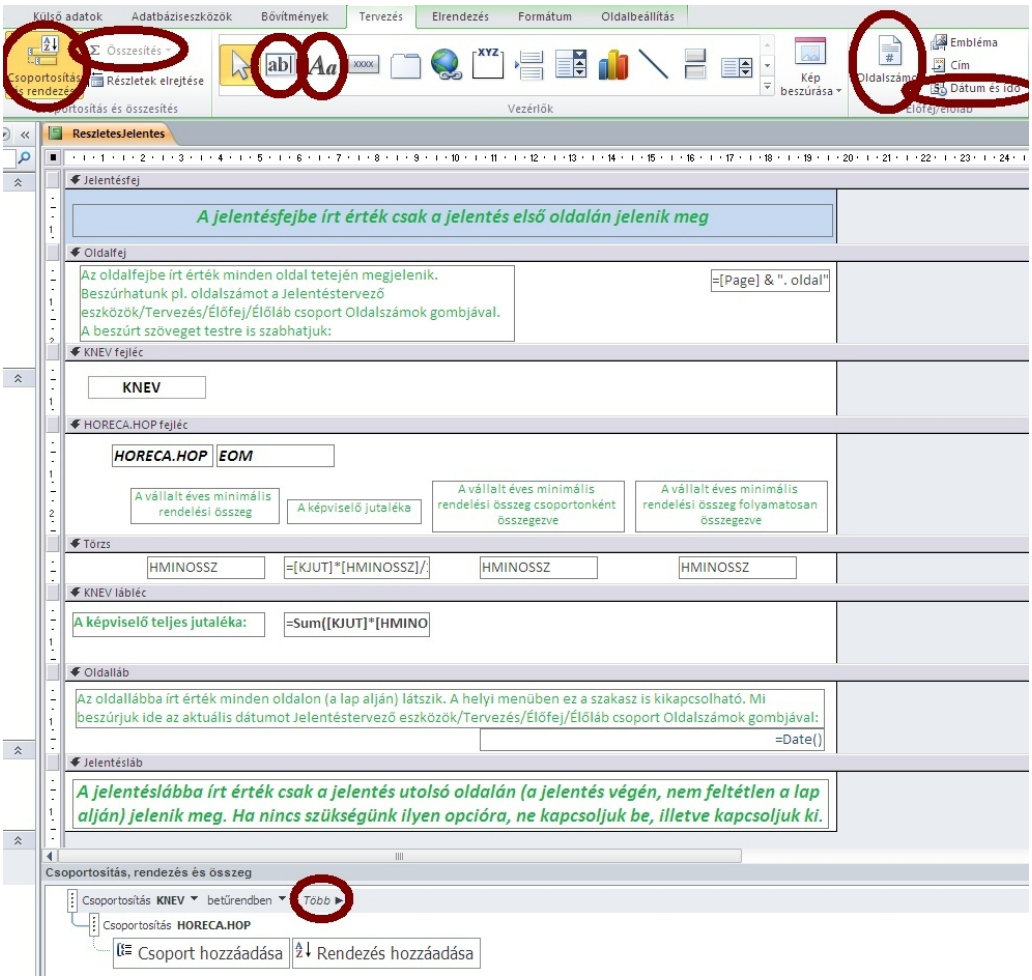
Kapcsoljuk be a *Jelentéstervező eszközök/Tervezés/Csoportosítás és rendezés* gombot, majd adjuk hozzá KNEV mezőt! Ez lesz az első csoportosítási szint. A különböző adatokat a képviselők neve szerint csoportosítva jelenítjük meg. A *KNEV fejléc* szakaszba felvisszük a képviselők nevét jelentő KNEV beviteli mezőt. Ez azt jelenti, hogy megjelenik egy képviselő neve – csak egyszer – majd alatta

a *Törzs* szakaszban felvett mezőkben csak azon adatok jelennek meg, amelyek az adott névhez köthetőek. A *Törzs* szakaszban viszont csak mezőnként egy-egy vezérlőelemet kell ehhez felvennünk függetlenül az adatok darabszámától. Ez csak elsőre meglepő. Valójában a KNEV fejlécbe írt adatok is sorban kiíródnak (bár egy vezérlőelemet csak egyszer veszünk fel itt is), csak széthúzva, megtöri őket a törzsbe felvett adatok kiírása. A KNEV csoportosítás után a HORECA.HOP szerint is hozzá kell adnunk egy csoportosítást a feladat szövege alapján. Az előző feladatban a HORECA.HOP és E_OPCIO.HOP használata közötti különbségre is rávilágítottunk. Jelen feladatban a HORECA.HOP helyett az E_OPCIO.HOP is használható lenne. A szükséges adat ez esetben is a HORECA.HOP felől áll elő, hiszen a törzsben a HORECA tábla adatai szerepelnek, ezek extra opció adatát kéri le a rendszer és eszerint csoportosít, vagy köt át az E_OPCIO tábla adatára. Ezért akár az extra opció megnevezése (EOM) mező is megfelelő választás lenne.

A csoportosítás megértéséhez váltsunk át az először készített mintajelentésre, majd lépjünk át tervező nézetbe és nézzük meg az adatok szervezését! Az összevetés és a csoportosítás megértése után a mintajelentést mentés nélkül bezárhatjuk.

Folytatjuk a feladatot. Csoportosítás esetén csak csoportfej jelenik meg, csoportláb nem. Az egyes képviselők összes jutaléka is kiírandó adat. Ezt képviselőnként egyszer szükséges kiírni, tehát a csoportosítás fej- vagy láblécében kell elhelyeznünk. Mi nem a képviselő neve mellé, hanem az egyes összegek felsorolása után, az adott képviselői adatok listázásának végén szeretnénk ezt elhelyezni. Ez a KJUT csoportlába, melyet be kell kapcsolni. Erre két lehetőségünk van. Megtehetjük, hogy a *Jelentéstervező eszközök/Tervezés/Összesítés* gomb alatt választjuk ki a szükséges függvényt. Ez csak az összesítendő mező kiválasztása után működik és mindig az adott mező szakasza felett közvetlenül következő szakaszban összesít. Ha például a törzsben található HMINOSSZ összegen állva kattintunk az említett gombra, akkor az összegzés alapértelmezetten a HORECA.HOP csoportlábba kerül. Ezért mi manuálisan adjuk hozzá a KNEV csoportlábat, majd helyezünk el benne egy (kötetlen) beviteli mező vezérlőelemet, melynek mező vagy kifejezés tulajdonságát fogjuk a megfelelő kifejezésre állítani. A csoportláb megjelenítéséhez ellenőrizzük, hogy a csoportosítás és rendezés gomb be van-e kapcsolva. Lépjünk a képernyő alsó részén található csoportosítás részhez, majd a *Csoportosítás KNEV* sor kijelölése után válasszuk a *Több* gombot. A kinyíló menüben váltsunk át a láblécszakasszal opcióra. A kért csoportlábléc megjelenik a képernyőn.

A törzsbe felvett mezők címkéjét az egy szinttel feljebb lévő csoportfejlécben helyeztük el, hogy az adott extra opcióhoz tartozó adatok esetén csak egyszer jelenjenek meg. Az előző instrukciók alapján az alábbi ábrán látható jelentéstervet kell elkésztenünk. A rácsot a jelentés helyi menüjében lekapcsoltuk, hogy még szemléletesebb képernyőképet tudjunk mellékelni.



46. ábra: 17. feladat – a jelentés tervező nézetben

Az ábrán a címke-vezérlők cím tulajdonságtípusában szereplő érték, illetve a beviteli mező vezérlők mező vagy kifejezés tulajdonságában szereplő érték látható. Az egyértelműség kedvéért a címke-vezérlők tartalma zöld színű. Az egyes vezérlők kisebbre is vehetőek, például a HORECA.HOP esetén érdemes lenne ezt tenni, illetve az EOM vezérlőjét közelebb rendezni. Akár még a vezérlők átfedése is elképzelhető. Mi azért választottuk az ábrán látható vezérlőméreteket, hogy a tervező nézetben bennük szereplő értékek egyértelműen olvashatóak legyenek, áttekinthető ábrát tudjunk mellékelni.

A két nem teljesen látható kifejezés:

```
=[KJUT]*[HMINOSSZ]/1000
```

illetve

```
=Sum([KJUT]*[HMINOSSZ]/1000)
```

Mindkét kifejezés vezérlőjének formátum tulajdonságát pénznem értékre, a tizedeshelyek tulajdonság értékét pedig 0-ra állítottuk, hogy a feladat előírásainak megfelelően Ft-ban jelenjenek meg az összegek.

A csoportosítás alapját képező mezőknél a rendezés alapból állítható, így adataink a KNEV, azon belül a HORECA.HOP mezők értékei alapján növekvő sorrendben jelennek meg. Lehetőségünk nyílna csak rendezést is hozzáadni a jelentéshez. Ki is próbálhatjuk! A csoportosítás és rendezés opcióinál (a képernyő alján) válasszuk a *Rendezés hozzáadása* gombot, majd a HKOTDAT mezőt és állítsuk be a legújabbtól a legrégebbiig opciót! Nézzük meg jelentés nézetben, mi történt a törzsben megjelenő adatok sorrendjével! Ezután törölhetjük ezt a rendezés-tesztbeállítást.

Nézzük meg a nyomtatási képet! (A jelentésnél ilyen nézetünk is van.) Néhány beállítást még el kell végeznünk. A különböző csoportok egymást követő adatsorait a rendszer különböző háttérszínnel látja el. Ez sok esetben igen hasznos, jelen esetben viszont kimondottan zavaró. Ennek lekapcsolásához a törzs, illetve a csoportok (KNEV, HORECA.HOP) láb- és fejléc szakaszának kijelölését követően a másodlagos háttérszín tulajdonságot kell átállítani. Mi a háttérszín és a másodlagos háttérszín tulajdonságokat is az *ablakok rendszerszíne* értékre (tulajdonképpen fehérre) állítottuk.

A beviteli mezők keret stílusa tulajdonságát *üresre* állítottuk. A KNEV mező vezérlője képez kivételt, a képviselők nevét keretezetten hagytuk.

A nyomtatási képen valószínűleg vízszintesen is széttöredezett az oldalunk. Az oldalbeállítás főbb opcióit a *Jelentéstervező eszközök/Oldalbeállítás* lapon érhetjük el. Az álló A4-es papír 21 cm széles, melybe a margó is beleértendő. A margók méretét az O*ldalbeállítás* menüpontban szabályozhatjuk. A jelentésnek a (21cm mínusz bal oldali margó mínusz jobb oldali margó) által adott eredményértékbe bele kell férnie vízszintesen, hogy szélességben egy oldalon maradhassunk. Ebben a vízszintes vonalzósáv, valamint az elrendezési nézet is segít. Elrendezési nézetben láthatóak a margók, amennyiben engedélyeztük ezt az *Oldalbeállítás* lapon. A vízszintes méret minden elemre vonatkozik. A jelentésfej szakasz kék háttere valószínűleg túlnyúlik a kívánt méreten és nem is tudjuk összébb tolni. Jelöljük ki a jelentésfejet és állítsuk az összenyomható tulajdonságát az *igen* értékre! Ezután a – nálunk kék színű háttérrel látható – jelentésfej is átméretezhetővé válik. Ha más elem is túllógna a jobb oldali margón és nem tudjuk átméretezni, akkor hasonlóképpen járjunk el.

Az egyes szakaszokon található feleslegesen nagy üres felületek előző feladatban is bemutatott hatása itt is érvényesül. Nyissuk szét a törzs szakaszt, a KNEV lábléc szakasz előtt legyen egy nagyobb üres, fehér rész. Nézzük meg a nyomtatási képet. A törzsben ismétlődő adatsorok széttolódtak, hiszen most az adatsor, üres rész, adatsor, üres rész… terjedelem ismétlődik. A KNEV láblécben nálunk látható kisebb üres sáv pont azt a célt szolgálja, hogy az újabb képviselő nevek előtt legyen némi üres terület, így is elválasztva az egyes képviselők adatait egymástól. A megfelelő eljárás tehát nem az egyes szakaszok egymáshoz tömörítése, hanem az egyes helyeken szükséges üres közök átgondolt beállítása.

A csoportonként összegezendő vállalt éves minimális rendelési összeg beviteli mező vezérlőjének futó összeg tulajdonságát állítsuk csoportonként értékre! A folyamatosan összegzendő vezérlőnél ugyanezt a tulajdonságot a folyamatosan értékre kell átváltani. Nézzük meg az eredményt! A csoportonként összegzett mező a legbelső (HORECA.HOP) csoport minden új értékénél újraindul, a folyamatosan összegzett mező a jelentés végéig aggregálódik.

Még egy beállítási lehetőséget nézzünk meg. A csoportosítás és rendezés beállításainál (a képernyő alján) valamelyik beállított csoportosítást kijelölve, majd a *Több* gombot választva lehetőségünk nyílik a csoport nyomtatási oldalon történő elhelyezését befolyásolni. Próbáljuk ki, válasszuk a KJUT csoportosítást, majd a maradjon egy oldalon a teljes csoport opciót! Nézzük meg a nyomtatási képen a változást! Ha valamely képviselő adatai eddig nem egy oldalon voltak, most egy új oldalon, együtt fognak szerepelni. Értelemszerűen, ha a csoport adatsora nem fér ki egy lapra, akkor ettől a beállítástól ez nem változik meg, többoldalas marad, viszont a csoport új oldalon fog kezdődni.

Mint látható, mind az űrlapoknál, mind a jelentéseknél rengeteg beállítási lehetőség van, a rendszer nagyon részletes paraméterezési lehetőségeket kínál. Érdemes az egyes beállítások hatását egyesével, külön-külön megnézni.

Nézzük meg alaposabban az elkészült jelentésünket! Csak négy képviselő szerepel, Clarice Starling nem. Ez a rekordforrásba beállított lekérdezésből fakad. A kérdéses képviselőhöz nem tartozik egyetlen szerződés sem, így bele sem kerül az inner join záradékú lekérdezésbe, azaz a jelentés forrásába. Tegyük bele Clarice Starling képviselőnket a jelentésbe!

A megoldás menete a left join lekérdezésnél írtakon alapszik. A KEPVISELO és HORECA táblák kapcsolatát úgy kell átalakítanunk, hogy a KEPVISELO tábla minden eleme belekerüljön a jelentés rekordforrás tulajdonságában található lekérdezésbe. Mentsünk! Mivel három tábláról van szó, ez így még nem elég, a rendszer nem is fogadja el a mentésünket. A lekérdezésbe bekerült az összes képviselő és a szerződések kapcsolódó rekordjai. Ezekhez azon extra opció neveket kötjük hozzá, amelyek kódjai a HORECA táblában szerepelnek. (Ne tévesszen meg minket, hogy most mindhárom kód szerepel a

HORECA táblában! Ha az E_OPCIO táblába felvennénk még egy OP04 elemet platina megnevezéssel, máris más lenne a helyzet. Ezt akár meg is tehetnénk, de tudnunk kell, hogy a HORECA táblában lehetetlen lenne ezt az opciót beállítani, mivel a HORECA.HOP mező jelenleg csak három értéket fogad a 2. feladat beállításai alapján, viszont a felvett új érték (platina) a Tervezett űrlapon létrehozott mezőben megjelenne. Azonban kiválasztása esetén hibaüzenetet kapnánk, mivel az adott vezérlőt úgy terveztük, hogy megjelenítendő értékekként az E_OPCIO.EOM mező értékeit szerepeltesse, a kiválasztott érték kódját viszont a HORECA.HOP mezőbe mentse, ahová az OP04 érték nem kerülhet be.) A zárójeles kitérő után tehát a teendő: a HORECA és E_OPCIO táblák kapcsolatát úgy kell átalakítanunk, hogy a HORECA tábla minden eleme belekerüljön a jelentés rekordforrás tulajdonságában található lekérdezésbe, az E_OPCIO táblából viszont csak azok a rekordok, ahol az illesztett mezők azonosak (right join).

Mentsük el az elkészült jelentést!

A következő feladat videójában megtekinthető az elkészült jelentés nyomtatási képe.

18. feladat – parancsgomb

Készítsen parancsgombot megfelelő ábrával az Urlap_Segeddel űrlapra, mely a ReszletesJelentes-re mutat!

A feladat megoldását az alábbi videón mutatjuk be:

19. feladat – menü készítése

Készítsen menüt, illetve helyi menüt az Urlap_Segeddel űrlaphoz!

A feladat megoldását az alábbi videón mutatjuk be:

20. feladat – navigáció, kapcsolótábla

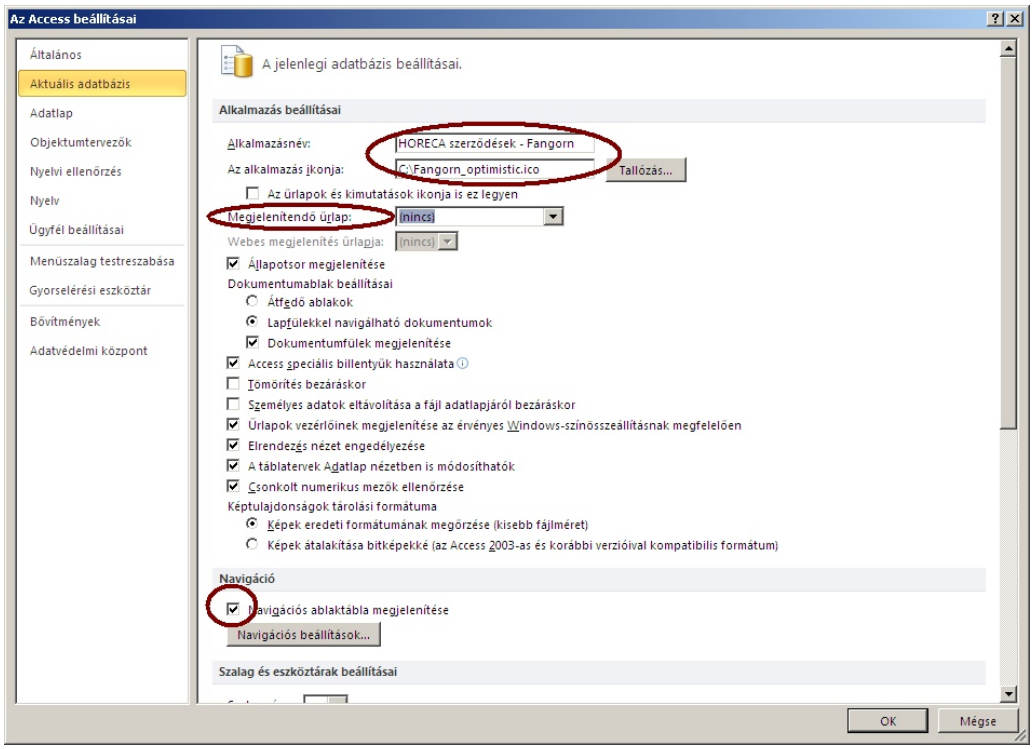
Hozzon létre egyéni navigációs panelt, mely az előzőleg készített űrlapokat és jelentést teszi elérhetővé!

A feladat megoldását az alábbi videón mutatjuk be:

+1 – az adatbázis további testre szabása

Az előző feladatokban látható volt, hogy a finomhangolásra számos eszköz áll rendelkezésre. Az objektumok és vezérlők rengeteg eseményre adhatnak makrókon, kifejezéseken vagy VBA (Visual Basic for Application) kódon keresztül egyéni választ.

Testre szabhatjuk a menüszalagot, eltüntethetjük a navigációs panelt. Így tulajdonképpen egy önálló saját programfelületet kapunk, ahol semmi sem adott. E felületen űrlapokkal számtalan különböző elképzelést valósíthatunk meg. Például jelszavas belépési felület is kialakítható űrlapként, de makrók, gombok és egyéb vezérlők segítségével akár egyéni menüfelület is kialakítható űrlap formában, mely az alkalmazás indulásakor a navigációs panel helyett megjelenik. A különböző nézetek letiltásával, egyéni helyi menükkel önálló működést tervezhetünk a rendszerben. Webes környezetben Sharepoint szerveren keresztül lehetőség van a Sharepoint jogosultságkezelési lehetőségeinek használatára is. Az Access tehát inkább keret, mintsem konkrét szoftver, a különböző mozaikok, funkciók kreatív használata mentén széles körű felhasználásra nyílik mód. A lehetőségek tárházát az alapok megértése után önálló, konkrét feladattervek mentén érdemes felderíteni, próbálgatni.



47. ábra: A Fájl/Beállítások menüpont lehetőségei

Az elkészült adatbázis

Az elkészült adatbázis állományt ide kattintva töltheti le. [images/HORECA_nyilvantartas.accdb]

Ha még nem jutott végig a feladatokon, mindenképpen nézze meg a 20. feladat beállításait. Ezek „inverzével” tudja az adatbázis meg nem jelenő objektumait visszahelyezni a navigációs panelre.

Irodalom

Bártfai, Barnabás. *Access 2007 zsebkönyv*. 2007. BBS-INFO.

Bhamidipati, Kishore. *SQL programozói referenciakönyv*. 1999. Panem.

Czenky, Márta. *Adatmodellezés példatár*. 2010. ComputerBooks Kiadó.

Codd, Edgar Frank Ted. *A Relational Model of Data for Large Shared Data Banks [http://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf] (Hozzáférés: 2012. július 26.)*. 1970. ACM. 13. (6). 377-387.

Date, Christopher J.. *An Introduction to Database Systems*. 2003. Addison Wesley.

Elmasri, Ramez és Navathe, Shamkant B.. *Fundamentals of Database Systems*. Fourth Edition. 2003. Addison-Wesley.

Fred, Rolland. *Adatbázisrendszerek*. 2002. Panem Kft.

Fodor, Ildikó. Tringer, Éva. *Adatbázis-kezelés*. 2003. Kossuth Kiadó Zrt.

Kiss, Jenő. *Adatbázis-kezelés*. 2007. Széchenyi István Egyetem.

Nagyné Lakatos, Eszter és Valentné Albert, Éva. *Adatbázis-kezelés FoxProban szoftverüzemeltetőknek*. 1998. Tótfalusi Tannyomda.

Siki, Zoltán. *Adatbáziskezelés és szervezés [http://www.agt.bme.hu/szakm/adatb/adatb.htm] (Hozzáférés: 2012. július 28.)*.

(szerző és évszám nélkül). *Adatbázis-kezelés Access XP-vel [http://www.kobakbt.hu/jegyzet/AccessXP.pdf] (Hozzáférés: 2012. augusztus 1.)*. Miniszterelnöki Hivatal, Informatikai Kormánybiztosság.

(szerző nélkül). *Az adatbázisok normalizálásnak alapjai [http://support.microsoft.com/kb/283878/hu] (Hozzáférés: 2012. július 27.)*. 2012.

(szerző nélkül). *Az adatbeviteli formátumok szabályozása beviteli maszkokkal [http://office.microsoft.com/hu-hu/access-help/az-adatbeviteli-formatumok-szabalyozasa-beviteli-maszkokkal-HA010096452.aspx?CTT=1] (Hozzáférés: 2012. július 29.)*. 2012.

(szerző nélkül). *Adattípusok és mezőtulajdonságok – Bevezetés [http://office.microsoft.com/hu-hu/access-help/adattipusok-es-mezotulajdonsagok-bevezetes-HA010341783.aspx] (Hozzáférés: 2012. július 28.)*. 2012.

(szerző nélkül). *A hivatkozási integritás megőrzése [http://office.microsoft.com/hu-hu/access-help/kapcsolat-letrehozasa-szerkesztese-es-torlese-HA010341606.aspx?CTT=1#_Toc269467499] (Hozzáférés: 2012. augusztus 3.)*. 2012.

(szerző nélkül). *A Kifejezésszerkesztő használata [http://office.microsoft.com/hu-hu/access-help/a-kifejezesszerkeszto-hasznalata-HA101812460.aspx?CTT=1] (Hozzáférés: 2012. július 28.)*. 2012.

(szerző nélkül). *A Lekérdezéstervező elrendezése – áttekintés (ADP) [http://office.microsoft.com/hu-hu/access-help/a-lekerdezestervezo-elrendezese-attekintes-adp-HP003083916.aspx] (Hozzáférés: 2012. július 29.)*. 2012.

(szerző nélkül). *Bevezetés az Access SQL nyelv használatába [http://office.microsoft.com/hu-hu/access-help/bevezetes-az-access-sql-nyelv-hasznalataba-HA010341468.aspx] (Hozzáférés: 2012. szeptember 3.)*. 2012.