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.