Adatbázisok

Gajdos Sándor

2015

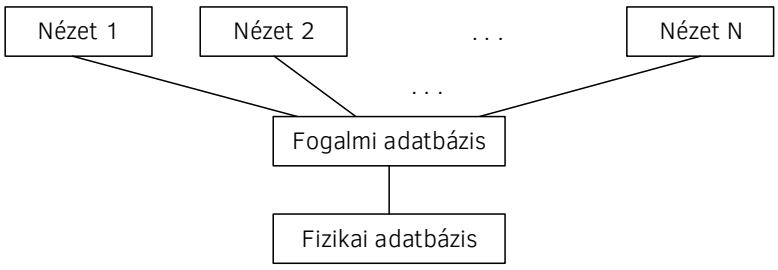
2. fejezet

Az adatbázis-kezelők felépítése

A mai adatbázis-kezelők bonyolult hardver-szoftver rendszerek. Komplexitásuk az operációs rendszerekével összemérhető, sőt, gyakran nagyobb annál. Egy ilyen rendszer megtervezése, implementálása és karbantartása nem egyszerű feladat, amelyre kifinomult módszerek léteznek. Ismertetésük túlmutat e jegyzet keretein, itt csak a legfontosabb modellezési, tervezést segítő elvek bemutatására van lehetőség.

Mint a mérnöki gyakorlatban olyan sok más helyen, itt is eredményes a rétegezési koncepció, vagyis egy *rétegmodell* (layered model) alkalmazása. Az alapgondolat az, hogy az eredeti problémát több részre kell bontani úgy, hogy az egyes részek egymásra épüljenek, de egymással csak minél kisebb felületen érintkezzenek. Jól ismert példa minderre a számítógép-hálózatok ISO-OSI modellje [11]. Hasonló modell, sőt modellek léteznek az adatbázis-kezelők számára is: a legegyszerűbb 3 rétegűtől kezdve a 7 rétegű modellig. Jelen jegyzetben részletesebben egy 3 rétegűvel ismerkedünk meg (2.1. ábra).

A legalsó réteg a *fizikai adatbázis* (physical database). Itt valósul meg az adatbázis adatainak a fizikai tárolókon való elhelyezése. Ide értjük azokat az adatstruktúrákat is, amelyekben a (fizikai) adattárolás megvalósul (ld. 3. fejezet). Ehhez a réteghez tartozó fogalmak: *kötet, állomány, blokk, track, szektor, vödrös hashing* (ld. 3.2. szakasz) stb.



2.1. ábra. Adatbázis-kezelők 3 rétegű architektúrája

Középen helyezkedik el a *fogalmi (logikai) adatbázis* (conceptual (logical) database). Ez nem más, mint a való világ egy darabjának leképezése, egy sajátos modell, ahogyan az adatbázis tükrözi a valóság egy részét. A fogalmi adatbázis van szorosabb kapcsolatban azzal, ahogyan az adatokat értelmezni kell. Pl. egy könyvtári adatbázisban ide tartoznak a következők: a kölcsönző személyek neve, kölcsönzőjegyének száma, egy kötet lelőhelye, ETO-száma, példányszáma, címe, szerzője, kiadója, értéke stb. A fogalmi adatbázishoz tartozó sémát *fogalmi (logikai) sémának* (conceptual (logical) schema) nevezik.

Nézet (view) az, amit és ahogy a felhasználó az adatbázisból lát. Ha az adatbázisnak több felhasználási lehetősége van, ezek mindegyikéhez külön nézet tartozhat. Ez lehet a felhasználók jogosítványaihoz kötött is. (Pl. a légitársaság egységes nyilvántartásából más adatok érdekesek, ha a pilóták szabadságolási tervét készítjük, és az adatok másik körére van szükségünk, ha egy gép utaslistáját akarjuk megtekinteni.) A nézetekhez tartozó sémákat gyakran *külső sémának* (external schema) is nevezik.

Minden jól megtervezett, a rétegezési koncepció alapján felépített rendszerben cél az, hogy a rétegek egymástól függetlenül megváltoztathatók, kicserélhetők legyenek, amennyiben a rétegek közötti interfészek változatlanok maradnak. Az adatbázis-kezelés világában ezt az *adatfüggetlenség* (data independence) elvének nevezik.

Kétféle adatfüggetlenségről lehet beszélni a háromrétegű modellben: a fizikai és a fogalmi adatbázis között értelmezhető *fizikai adatfüggetlenségről*, ill. a fogalmi adatbázis és a nézetek között értelmezhető *logikai adatfüggetlenségről*.

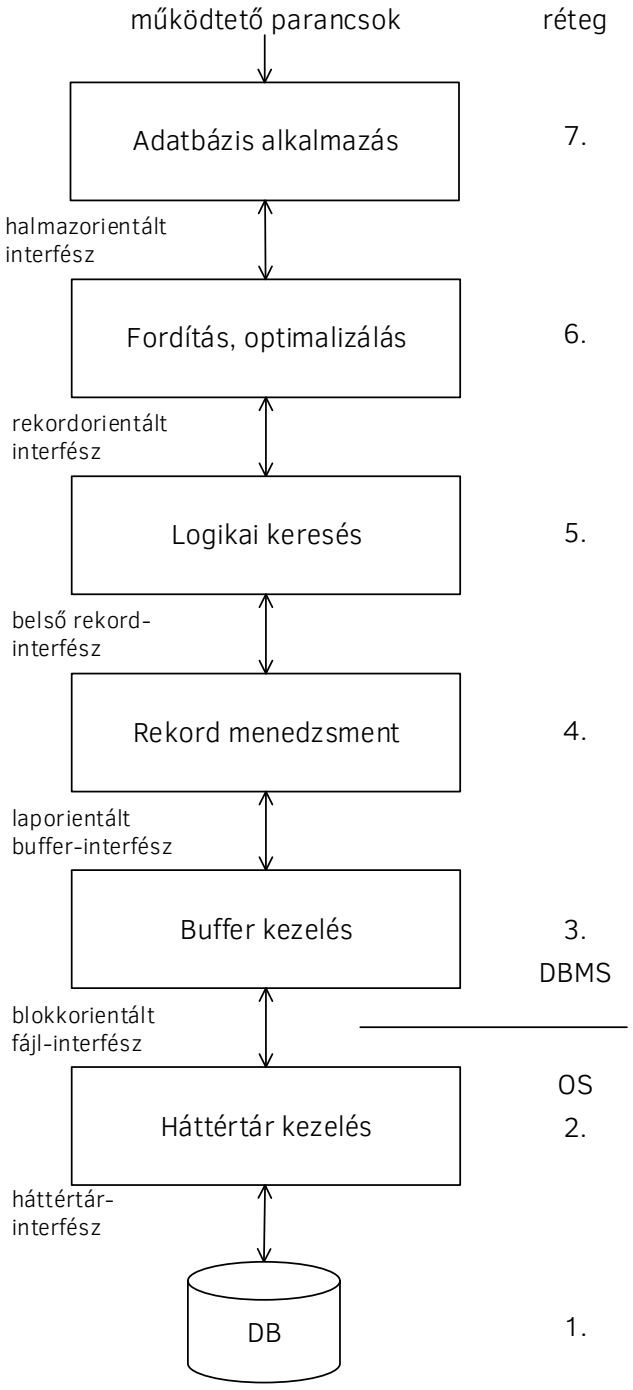
A *fizikai adatfüggetlenségen* (physical data independence) (kb. eszközfüggetlenségen) azt értjük, hogy a fizikai szinten, a fizikai működés sémáiban véghezvitt változások nem érintik a fogalmi (logikai) adatbázist. Ha ez teljesül (gyakorlatilag mindig), akkor a fizikai adathordozó egy teljesen eltérő fizikai paraméterekkel rendelkezőre is kicserélhető (pl. meghibásodás, technikai fejlődés stb. miatt), vagy az állományszervezés módja megváltoztatható anélkül, hogy az adatbázisban bármilyen logikai változás érzékelhető lenne (a rendszer teljesítőképessége, válaszidejei azonban jelentősen változhatnak).

Logikai adatfüggetlenségről (logical data independence) akkor beszélünk, ha a logikai adatbázis megváltozása nem jár az egyes felhasználásokhoz-felhasználókhoz tartozó nézetek megváltozásával. Ez az elvárás már nem teljesül minden esetben.

Illusztrációképpen bemutatjuk a 2.2. ábrán az adatbázis-kezelőnek és környezetének egy tipikus, hétrétegű modelljét.

2.1. A fejezet új fogalmai

adatbázis nézet (view), modell, rétegmodell, fizikai adatbázis, logikai (fogalmi) adatbázis, külső séma, logikai adatfüggetlenség, fizikai adatfüggetlenség



2.2. ábra. Adatbázis-kezelő (és környezete) statikus 7 rétegű modellje

4. fejezet

A fogalmi (logikai) adatbázis

Ebben a fejezetben a 2.1. ábrán megismert adatbázis-modell középső, logikai részét vizsgáljuk meg részletesebben.

4.1. Adatmodellek, modellezés

Amikor egy *adat*bázist létrehozunk, a cél az, hogy benne a való – vagy ritkábban egy kitalált – világ *adat*ait tároljuk úgy, hogy belőle a való (kitalált) világról *információkat* nyerhessünk ahelyett, hogy a valóságból kelljen ugyanazt az információt megszerezni. Általában nincsen mód egy adott probléma- (téma- vagy jelenség-) körrel kapcsolatos összes adat tárolására, így adatoknak csak egy meghatározott, szűk körét kezelhetjük. A tárolandó adatok kiválasztásánál klasszikus modellezési szempontok érvényesülnek, azaz a vizsgálat szempontjából fontosnak tartott jellemzőket tároljuk, a többit elhanyagoljuk (jellemző alatt itt egyaránt értünk tulajdonságokat és kapcsolatokat is). Így az adatbázis a világ egy darabjának egy leegyszerűsített képét adja vissza.

Amikor ezt a képet elkezdjük kialakítani, követhetünk bizonyos konvenciókat, ami számos előnnyel járhat. A konvenciók egy része arra vonatkozik, hogy milyen formában, milyen kapcsolatok kialakítását támogassuk az adataink között és hogy milyen műveleteket engedjünk meg az adatainkon. Így ún. adatmodelleket hozunk létre. Természetesen, a konvenciókhoz való alkalmazkodás járhat hátrányokkal is, ez esetben megfontolandó egy teljesen egyedi adatmodell megalkotása.

Egy *adatmodell* (data model) tehát hagyományosan két részből áll:

1. formalizált jelölésrendszer adatok, adatkapcsolatok leírására
2. műveletek az adatokon.

Az adatmodell tulajdonságai alapvetően meghatározzák az azt használó adatbázis tulajdonságait. A felhasználó számára pedig az adatbázisnak az egyik legfontosabb jellemzője az a forma, amelyben a tárolt adatok közötti összefüggések ábrázolva vannak. Az ábrázolás alapegysége a rekord, ill. a rekordtípus (vagy egy ezzel

analóg, de esetleg másképpen nevezett konstrukció). Mivel egy adatbázis struktúráját jelentős részben a rekordtípusok közötti kapcsolatok határozzák meg, ezért az adatmodelleket aszerint osztályozzuk, hogy a rekordtípusok között milyen kapcsolatok definiálása megengedett, azaz a felhasználó szempontjából miként valósul meg az adatok közötti kapcsolatok ábrázolása.

A hálós adatmodellnél (ld. 7. fejezet) a rekordtípusok között (pl. mutatók segítségével) tetszőleges függvényszerű kapcsolatokat szervezhetünk. A relációs adatmodellnél a kapcsolatok kialakítására nincs külön strukturális elem, magukat a kapcsolatokat is relációkkal ábrázoljuk (ld. 5. fejezet). Az objektumorientált adatmodell (ld. 8. fejezet) objektumokat tartalmaz, amelyek között változatos típusú kapcsolatokat hozhatunk létre, ezért sok szempontból a hálós adatmodellhez hasonlatos.

Az adatmodell tehát meghatározza, hogy az adatbázisban az adatok milyen struktúrában tárolódnak, és milyen mechanizmusokon keresztül lehet az adatokhoz hozzáférni. Így az adatbázis-kezelő rendszer legalapvetőbb tulajdonságait rögzíti. Egy adatbázis-kezelő rendszer ezért csaknem mindig egyetlen adatmodellnek megfelelően működik.

4.2. Egy majdnem-adatmodell: az egyed-kapcsolat modell

Az *egyed-kapcsolat* (entity-relationship, ER) modell nem tekinthető a fenti értelemben adatmodellnek, mert nincsenek benne adatműveletek definiálva.

4.2.1. Az ER-modell elemei

Az ER-modell elemei:

- *egyedtípusok*;
- *attribútumtípusok*;
- *kapcsolattípusok*.

Természetesen, a *típusokhoz* mindenütt tartoznak konkrét *példányok* (*eset*, *előfordulás*) is, de maga a modellezés a típusok szintjén történik. Összhangban azzal, amit általában típusnak nevezünk, a típus itt is a konkrétan létező – de hasonló – egyedek, tulajdonságok, kapcsolatok absztrakciója. Az egyedek (tulajdonságok, kapcsolatok) bizonyos közös jegyek alapján halmazokba rendeződnek. Egy-egy halmaz neve az egyed (tulajdonság, kapcsolat) típusa, a halmazok elemei pedig a példányok.

4.2.1.1. Entitások

Definíció – *egyed,* entitás *(entity).* A valós világban létező, logikai vagy fizikai szempontból saját léttel rendelkező dolog, amelyről adatokat tárolunk.

Megjegyzés. Ennek megfelelően az egyedeknek megkülönböztethetőknek kell lenniük, mint ahogyan a matematikai értelemben definiált halmazok elemei is azok. Entitás lehet (!) egy autó, egy személy, egy szerződés, de még a szeretet is, hiszen *megfelelő attribútumok megválasztásával* az autók, személyek stb. megkülönböztethetővé tehetők, azaz „saját létet rendelhetünk hozzájuk". Ugyanakkor általában nem tekinthető entitásnak egy tojás vagy egy hangya, mivel a tojás- vagy a hangya-példányok rendszerint nem különböztethetők meg.

Definíció – tulajdonság *(property).* Az entitásokat jellemzi, amelyen vagy amelyeken keresztül az entitások megkülönböztethetők.

Megjegyzés. Valójában az entitások definiálása modellezési kérdés. A modellalkotón múlik, hogy milyen tulajdonságokat rendel hozzá egy-egy entitáshoz, így biztosítja-e azok megkívánt szintű megkülönböztethetőségét. Elképzelhető, hogy egy tudományos adatbázisban éppen hangyák adatait kell tárolni, amelyekkel különböző kísérleteket végeztek. Ekkor a hangyák *megkülönböztethetővé tehetők* egy mesterségesen hozzájuk rendelt, egyediséget biztosító *attribútummal* (attribute) – a gyakorlatban pl. az elkülönített tárolásuk segítségével –, így a hangyák is entitásokká válhatnak.

Definíció – egyedhalmaz *(entity set).* Az azonos attribútumtípusokkal jellemzett egyedek összessége.

Az entitások közös attribútumtípusait zárójelben szokás az entitáshalmaz neve után felsorolni.

Példa.

- `EMBER(név, szül_dátum, anyja_neve, szeme_színe, személyi_szám)`
- `SZERZŐDÉS(cég1, cég2, dátum, hely, szerződés_tárgya, érték, telj_határidő)`

4.2.1.2. Kapcsolatok

A valóságban az egyedek ritkán léteznek elszigetelten, egymástól függetlenül. Tipikus az, hogy valamilyen kapcsolatban állnak egymással: az emberek cégeknél *dolgoznak*, szerződéseket *írnak alá*, egymással rokoni kapcsolatban lehetnek (pl. *testvére valakinek*). Ezeket a tényeket kifejezhetjük, ha az entitáshalmazok között kapcsolat típusokat definiálunk. Természetesen ezeknek a meghatározása szintén

modellezési kérdés: az adott feladat dönti el, hogy egy konkrét adatbázisban milyen kapcsolat típusok definiálása szükséges.

Definíció – kapcsolat *(relationship)*. Entitások névvel ellátott viszonya.

Formálisan egy kapcsolat típus nem más, mint entitás típusok névvel ellátott sorozata.

Példa.

- *DOLGOZIK*: EMBER, CÉG
  Ez a bináris kapcsolattípus azt fejezheti ki, hogy valaki egy cégnél dolgozik.
- *ALÁÍR*: EMBER, CÉG, SZERZŐDÉS
  Ez a ternáris (hármas) kapcsolattípus azt fejezheti ki, hogy egy személy egy cég nevében egy szerződést aláírt.
- *TESTVÉRE*: EMBER, EMBER
  Ez a bináris kapcsolat típus azt fejezheti ki, hogy az egyik ember testvére egy másiknak. Ennek a kapcsolattípusnak egy példánya – egy konkrét kapcsolat – pl. azt fejezheti ki, hogy Kis Géza testvére Kis Antalnak.

A kapcsolatok igen sokfélék lehetnek. Fontos szempont, hogy hány entitáshalmaz között teremtenek kapcsolatot, vagy hogy egy kiválasztott példány hány másikkal lehet kapcsolatban. Ezen belül érdekes lehet, hogy egy kiválasztott példányhoz mindig tartozik-e egy vagy több másik példány, ha igen, akkor mennyi a kapcsolódó egyedek minimális, maximális száma stb. A kapcsolatok teljes mélységű jellemzése gyakran szükségtelen, mi is csak olyan mélységben fogjuk megtenni, amit a tervezett felhasználás indokol.

Megjegyzés. A mindennapi gyakorlatban (sajnos) rendszerint elfeledkezünk a típusok (halmazok) és a konkrét példányok megkülönböztetéséről. Igen gyakran emlegetünk entitást, kapcsolatot akkor is, amikor valójában entitás- vagy kapcsolattípusról/halmazról van szó. Ez megtehető általában, mert a szövegkörnyezet miatt többnyire nem okoz félreértést ez a pontatlanság. Engedve a szokásnak, a továbbiakban nem hangsúlyozzuk a különbséget, ha ez kétértelműséget nem okoz.

4.2.1.2.1. Kapcsolatok funkcionalitása (kardinalitás) Említettük, hogy a kapcsolatok különbözhetnek pl. abban is, hogy egy entitáshalmaz egy eleméhez egy másik entitáshalmaznak hány elemét rendelik hozzá. A legegyszerűbb csoportosításban egy-egy, egy-több vagy több-több kapcsolatról beszélünk.

Definíció – egy-egy kapcsolat *(one-to-one relationship)*. Olyan (bináris) kapcsolat, amelyben a résztvevő entitáshalmazok példányaival egy másik entitáshalmaznak legfeljebb egy példánya van kapcsolatban.

Példa.

- *HÁZASSÁG*: EMBER, EMBER
- *FŐNÖK*: OSZTÁLY, EMBER

Megjegyzés *(1)*. Vegyük észre, hogy egy kapcsolat funkcionalitásának meghatározása is modellezési kérdés. Általában (Magyarországon) igaz ugyanis, hogy a *HÁZASSÁG* egy-egy kapcsolat, hiszen egy emberhez (egy időben) legfeljebb egy másik embert rendel hozzá. Elégtelen lenne azonban a valóságnak ezen szintű modellezése, ha az adatbázisunknak olyan iszlám országban is működnie kellene, ahol a többnejűség is megengedett.

Megjegyzés *(2)*. Egy-egy kapcsolatok még abban is különbözhetnek, hogy az egyik entitáshalmaz példányai minden esetben kapcsolatban vannak-e egy másik entitáshalmaz egy példányával, vagy nem feltétlenül tartozik hozzá egy másik példány.

Az előbbi helyzetre jellemző a *FŐNÖK* kapcsolat, hiszen általában minden osztálynak pontosan egy főnöke van. Ellenkező irányban mindez már nem feltétlenül igaz, hiszen az EMBER entitáshalmaznak nem minden példánya kell, hogy főnöke legyen valamely osztálynak.

A *HÁZASSÁG* kapcsolatban szereplő entitáshalmazban lehetnek olyan személyek, akik egyedülállók, így egyik irányban sem teljesül, hogy egy kiválasztott példányhoz feltétlenül tartozik is egy másik példány.

Ezek a megfontolások a kapcsolatok mélyebb analízisének lehetőségeire utalnak.

Definíció – több-egy kapcsolat *(many-to-one relationship)*. Egy *K*: E1, E2 kapcsolat több-egy, ha E1 példányaihoz legfeljebb egy E2-beli példány tartozik, míg E2 példányai tetszőleges számú E1-beli példányhoz tartoznak.

Példa. *TANUL*: DIÁK, OSZTÁLY

Itt tételezzük azt fel, hogy egy véletlenszerűen kiválasztott diák általában egy osztályban tanul (egyidejűleg), de egy meghatározott osztályba számos diák jár (egyidejűleg).

Definíció – több-több kapcsolat *(many-to-many relationship)*. Egy kapcsolat több-több funkcionalitású, ha nem több-egy egyik irányban sem.

Példa. *TAN*: DIÁK, TANÁR

Ez a kapcsolat azt fejezheti ki, hogy diákok és tanárok „tanítja – tanul nála” viszonyban lehetnek egymással. A kapcsolat több-több funkcionalitású, mert egy

tanár több diákot is taníthat, és egy diák több tanárnál is tanulhat.

Az adatbázis-kezelésben a több-egy (egy-több) kapcsolatok kitüntetett jelentőségűek, mert viszonylag egyszerűen ábrázolhatók, ugyanakkor elegendően általánosak, kifejezőek is.

4.2.2. Kulcs

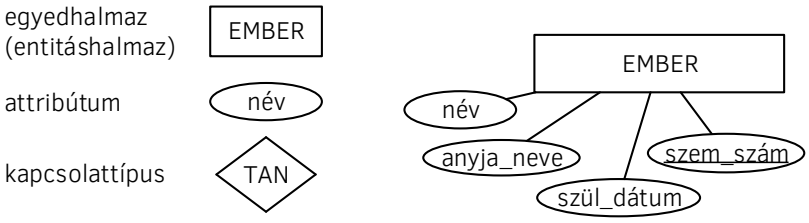
Definíció – kulcs *(key)*. Az ER-modellezésnél az attribútumoknak azt a halmazát, amely az entitás példányait egyértelműen azonosítja *kulcsnak* nevezzük.

Példa. Az `EMBER` entitáshalmaz elemeit egyértelműen azonosítja a `(név, szül_dátum, anyja_neve)` attribútumhármas, vagy a `személyi_szám` attribútum. Az `EMBER` entitáshalmaznak tehát két kulcsa is van.

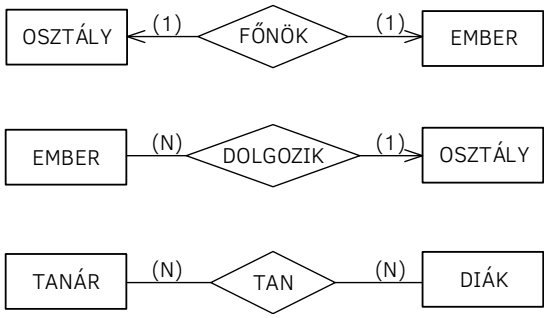
Minden egyedhalmaznak legalább egy kulcsa mindig van, hiszen az egyedeknek megkülönböztethetőknek kell lenniük. Ehhez pedig az attribútumok teljes halmaza elegendő, tehát az attribútumok teljes halmaza mindig kulcs. A kulcs attribútumait hagyományosan aláhúzással jelöljük.

4.2.3. Az ER-modell grafikus ábrázolása: ER-diagram

Bár az előbbiekben bevezetett formális jelölésrendszer elegendő az ER-modell megadására, a gyakorlatban elterjedten használnak (különböző) grafikus megjelenítési formákat is. Mi most az eredeti jelölésrendszert mutatjuk be.



4.1. ábra. Az ER-diagram elemei



4.2. ábra. Kapcsolatok funkcionalitásának egy ábrázolása az ER-diagramoknál

Példa. A Nekeresdi általános Biztosítónak számos kirendeltsége működik szerte Nekeresdország városaiban, néhányban több is. Minden kirendeltségnek külön kódszáma is van, amely egyértelműen azonosítja őket. A kirendeltségeken többen is dolgoznak, de egy alkalmazott egy évben csak egy kirendeltségnél vállal munkát. A dolgozókat kódjuk egyértelműen meghatározza, de tárolni kell róluk még a nevüket, beosztásukat és fizetésüket is. Az alkalmazottak időnként munkahelyet változtatnak – de mindig csak január 1-jei dátummal –, és a Nekeresdi általános Biztosítón belül másik kirendeltséghez mennek dolgozni.

A leírás alapján pl. az alábbi ER-modellt alkothatjuk.

Entitáshalmazok:

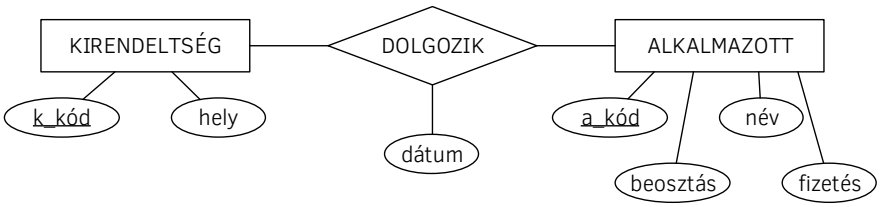
- `KIRENDELTSÉG(k_kód, hely)`
- `ALKALMAZOTT(a_kód, név, beosztás, fizetés)`

Kapcsolattípus:

- *`DOLGOZIK`*`: KIRENDELTSÉG, ALKALMAZOTT; dátum`

Megjegyzés. A dátum attribútum – ami egy évszám, és azt fejezi ki, hogy egy adott évben mely alkalmazottak dolgoztak egy adott kirendeltségen – nem tartozik egyedül sem a kirendeltség sem az alkalmazott entitásokhoz, mindig csak a kettőhöz együtt. Az ER-modell azonban nem engedi meg ilyen attribútumok definiálását. Így kénytelenek vagyunk a „dátum”-ot önmagában entitáshalmazzá tenni és a két érintett entitáshalmaz között értelmezett *`DOLGOZIK`* kapcsolatba belevonni.

Mindezt a 4.3. ábra ER-diagramján is ábrázolhatjuk. Mivel a diagramot a fenti ER-modell – és nem az eredeti leírás – alapján alkottuk, ezért nem látszik rajta, hogy a *`DOLGOZIK`* kapcsolathalmaz 1:N funkcionalitású. Természetesen ezt egy jobb, pontosabb diagramon akár ábrázolhatnánk is.

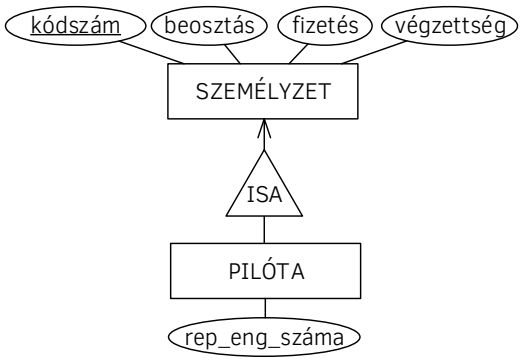


4.3. ábra. ER-diagram a fenti ER-modell alapján

Az ER-diagramok eszköztárát évtizedek alatt sokan, sokféle módon terjesztették ki, leginkább annak érdekében, hogy a gyakorlatban felmerülő számos különböző jelentést hordozó kapcsolatokat meg lehessen különböztetni. Így jöttek létre a különböző EER-diagramok (Extended ER). Az alábbiakban ennek néhány elemét mutatjuk be.

Gyakori az a modellezési szituáció, amikor egy entitáshalmaz minden eleme rendelkezik egy másik (általánosabb) entitáshalmaz attribútumaival, de azokon kívül még továbbiakkal is (specializáció). Ez a viszony a kapcsolatok egy sajátos típusával, az ún. „isa”[1] kapcsolattal írható le.

Példa.



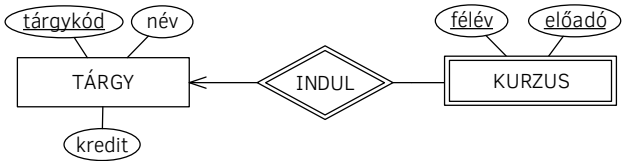
4.4. ábra. Specializáció ábrázolása ER-diagramon

Az isa kapcsolatnak az objektumorientált modelleknél kitüntetett szerepe van.

Szintén gyakori, hogy a modellezés során egy entitáshalmaznak nem tudunk kulcsot meghatározni, hanem az egyedek azonosításához valamely kapcsolódó egyed(ek)re is szükség van. Ebben az esetben *gyenge egyedhalmazról* (weak entity set) beszélünk. A gyenge egyedhalmaz identitását egy (vagy ritkán több) ún. *tulajdonos egyedhalmaz* (owner entity set) biztosítja, amely a gyenge egyedhalmazzal több-egy kapcsolatban áll. A kapcsolat neve *determináló kapcsolat* (identifying relationship).

[1] is a: angol.

A gyenge egyedhalmaz és determináló kapcsolatának szokásos jelölése a 4.5. ábra diagramján látható, ahol a `KURZUS` egyedhalmaznak nincs kulcsa, mert pl. a 2012/2013/1. félévben Gipsz Jakab több kurzust is vezethet. A kurzusokhoz a megfelelő tárgykódokat hozzárendelve lesznek az egyes kurzusok mint entitások egyértelműen megkülönböztethetők. A `KURZUS` tehát gyenge egyedhalmaz, az *`INDUL`* a determináló kapcsolata, ezért a `KURZUS` példányok egyedisége csak a `TÁRGY` példányaival együtt biztosítható.



4.5. ábra. Gyenge egyedhalmaz és a determináló kapcsolat ábrázolása ER-diagramon

A gyenge egyedhalmaz példányait (a hozzájuk tartozó tulajdonos egyedhalmaz kulcs attribútumaival együtt) egyértelműen megkülönböztető attribútumokat a kulcs attribútumokhoz hasonlóan aláhúzással jelöljük.

4.3. A fejezet új fogalmai

modell, modell az adatokról, adatmodell, formális jelölésrendszer, egyed-kapcsolat (ER) modell, ER-diagram, entitás, kapcsolat, tulajdonság (attribútum), egyedtípus, kapcsolattípus, tulajdonságtípus, kapcsolat fokszáma, kapcsolat funkcionalitása (kardinalitása), isa kapcsolat, gyenge egyed, determináló kapcsolat, kulcs