Fejezetek az adatbázisrendszerek elméletéből

A válogatott fejezetek Ramez Elmasri és Shamkant B. Navathe *Fundamentals of Database Systems* című könyve alapján készültek.

Kósa, Márk

Pánovics, János

A tananyag a TÁMOP-4.1.2-08/1/A-2009-0046 számú Kelet-magyarországi Informatika Tananyag Tárház projekt keretében készült. A tananyagfejlesztés az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósult meg.

Nemzeti Fejlesztési Ügynökség http://ujszechenyiterv.gov.hu/ *06 40 638-638*

2011

Tartalom

Az ábrák listája

4.9. EER koncepcionális séma az EGYETEM adatbázishoz.
4.10. EER séma a KIS_REPÜLŐTÉR adatbázishoz.
5.1. Egy HALLGATÓ reláció attribútumai és rekordjai.
5.2. Az 5.1. ábra HALLGATÓ relációja a rekordok más sorrendjében.
5.3. Sémadiagram a VÁLLALAT relációs adatbázissémához.
5.4. Egy, a VÁLLALAT adatbázissémához tartozó lehetséges adatbázis-állapot
5.5. Hivatkozási integritási megszorítások megjelenítése a VÁLLALAT relációs adatbázissémán.
5.6. A REPÜLŐGÉP_MENETREND relációs adatbázisséma.
6.1. Szelekció és projekció műveletek eredményei. (a) $\sigma_{(Osz=4\ \textbf{AND}\ Fizetés>325000)\ \textbf{OR}\ (Osz=5\ \textbf{AND}\ Fizetés>390000)}(DOLGOZÓ)$. (b) $\pi_{Vnév,\ Knév,\ Fizetés}(DOLGOZÓ)$. (c) $\pi_{Nem,\ Fizetés}(DOLGOZÓ)$.-
6.2. Egy műveletsorozat eredménye. (a) $\pi_{Vnév,\ Knév,\ Fizetés}(\sigma_{Osz=5}(DOLGOZÓ))$. (b) Közbenső relációk használata és az attribútumok átnevezése.
6.3. Az EREDMÉNY ← EREDMÉNY1 ∪ EREDMÉNY2 egyesítés (unió) művelet eredménye.
6.4. Az egyesítés (unió), metszet és különbség (kivonás) műveletek. (a) Két uniókompatibilis reláció. (b) HALLGATÓ ∪ OKTATÓ. (c) HALLGATÓ ∩ OKTATÓ. (d) HALLGATÓ − OKTATÓ. (e) OKTATÓ − HALLGATÓ.
6.5. A Descartes-szorzat (keresztszorzás) művelet.
6.6. Az OSZTÁLYVEZETŐ ← OSZTÁLY $\bowtie_{Vez\_szsz\ =\ Szsz}$ DOLGOZÓ join művelet eredménye.
6.7. Relációs adatbázisséma egy KÖNYVTÁR adatbázishoz.
7.1. ER koncepcionális séma a VÁLLALAT adatbázishoz
7.2. A VÁLLALAT ER séma relációs adatbázissémára történő leképezésének eredménye
7.3. A leképezés néhány lépésének bemutatása. (a) *Egyedtípusból képzett* relációsémák az első lépés után. (b) Egy új, *gyenge egyedtípusból képzett* relációséma a második lépés után. (c) *Kapcsoló* relációséma az 5. lépést követően. (d) Többértékű attribútumot reprezentáló relációséma a 6. lépés után.
7.4. A 3.16. (a) ábra *n*-edfokú SZÁLLÍT kapcsolattípusának leképezése.
7.5. A specializációk és generalizációk leképezésének lehetőségei. (a) A 4.4. ábrán látható EER séma leképezése a 8A opcióval. (b) A 4.3. (b) ábrán látható EER séma leképezése a 8B opcióval. (c) A 4.4. ábrán látható EER séma leképezése a 8C opcióval. (d) A 4.5. ábrán látható EER séma leképezése a logikai típusú Gyjelző és Vjelző típus mezőkkel.
7.6. A 4.7. ábra EER specializációs hálójának leképezése többféle opció használatával.
7.7. A 4.8. ábrán látható EER kategóriák (unió típusok) leképezése relációsémákra.
7.8. ER séma a HAJÓKÖZLEKEDÉS adatbázishoz.
8.1. Egy egyszerűsített VÁLLALAT relációs adatbázisséma.
8.2. Példa adatbázis-állapot a 8.1. ábrán látható relációs adatbázissémához.
8.3. Két, módosítási anomáliákkal terhelt relációséma
8.4. Példa állapotok a 8.2. ábra relációira alkalmazott természetes összekapcsolással kapott DOLG_OSZT és DOLG_PROJ relációkhoz. Ezek hatékonysági szempontok miatt alaprelációként tárolhatók.
8.5. A 8.3. (b) ábra DOLG_PROJ relációjának különösen rossz terve. (a) A DOLG_HELYSZÍNEK és a DOLG_PROJ1 relációsémák. (b) A 8.4. ábra DOLG_PROJ relációjának a DOLG_HELYSZÍNEK és DOLG_PROJ1 relációkra történő vetítésének az eredménye.
8.6. A 8.5. ábra DOLG_PROJ1 és DOLG_HELYSZÍNEK relációinak a szaggatott vonalak fölötti rekordjaira alkalmazott természetes összekapcsolás eredménye. A kapott álrekordokat csillaggal jelöltük meg.
8.7. A TANÍT egy relációállapota a *lehetséges* Jegyzet → Kurzus funkcionális függéssel. Az Oktató → Kurzus azonban ki van zárva.
8.8. Normalizálás 1NF-be. (a) Egy relációséma, amely nincs 1NF-ben. (b) Az OSZTÁLY reláció példa állapota. (c) Ugyanazon reláció 1NF változata redundanciával.
8.9. Beágyazott relációk normalizálása 1NF-be. (a) A DOLG_PROJ reláció sémája a Projektek *beágyazott reláció* attribútummal. (b) A DOLG_PROJ reláció példa állapota mutatja a beágyazott

A táblázatok listája

1. fejezet - Adatbázisok és adatbázis-felhasználók

Tartalom

Bevezetés

Az adatbázisok és az adatbázis-technológia nagy hatással vannak a számítógépek elterjedésére. Azt mondhatjuk, hogy az adatbázisok kritikus szerepet játszanak szinte minden olyan területen, ahol számítógépeket használunk, beleértve az üzletet, az elektronikus kereskedelmet, a mérnöki tudományokat, az orvostudományt, a jogot, az oktatást és a könyvtártudományt. Az *adatbázis* szó használata annyira elterjedt, hogy annak a definiálásával kell kezdenünk, mi is az az adatbázis. Az első definíciónk meglehetősen általános.

Az **adatbázis** egymással kapcsolatban álló adatok gyűjteménye. **Adat** alatt olyan ismert tényeket értünk, amelyek feljegyezhetők, és amelyeknek implicit jelentésük van. Ilyenek például az ismerőseink nevei, telefonszámai és címei. Ezeket az adatokat feljegyezhetjük egy indexelt névjegyzékbe, vagy tárolhatjuk merevlemezen is egy személyi számítógépen egy olyan szoftverrel, mint például a Microsoft Access vagy Excel. Az ilyen implicit jelentéssel bíró, egymással összefüggő adatok gyűjteménye az adatbázis.

Az adatbázisnak ez a definíciója meglehetősen általános; egymással összefüggő adatoknak tekinthetjük például az ezen az oldalon található szavak gyűjteményét, amelyek ezáltal egy adatbázist alkotnak. Az *adatbázis* kifejezést viszont általában ennél korlátozottabb értelemben használhatjuk. Az adatbázis a következő implicit tulajdonságokkal rendelkezik:

- Az adatbázis a valós világ valamely részét reprezentálja, amelyet néha **minivilágnak** vagy **a modellezés tárgyának** nevezünk. A minivilágban bekövetkező változások megjelennek az adatbázisban.
- Az adatbázis adatok logikailag összetartozó gyűjteménye a benne rejlő jelentéssel együtt. Az adatok egy véletlenszerű összességét nem tekinthetjük adatbázisnak.
- Az adatbázist egy konkrét céllal tervezzük és építjük, továbbá konkrét céllal tárolt adatokkal töltjük föl. Jól meghatározott felhasználói csoport használja jól meghatározott, számukra értelmes céllal.

Más szóval az adatbázisnak van valamilyen forrása, ahonnan az adatok származnak, valamilyen szinten kölcsönhatásban áll a valós világ eseményeivel, és van egy közönsége, amelyet aktívan érdekel a tartalma. Az adatbázis végfelhasználói végrehajthatnak üzleti tranzakciókat (például egy vevő vásárol egy kamerát), vagy olyan események történhetnek (például egy alkalmazottnak gyereke születik), amelyek az adatbázisban tárolt információk megváltozását okozzák. Ahhoz, hogy egy adatbázis minden pillanatban pontos és megbízható legyen, hűen kell tükröznie az általa reprezentált minivilágot; ezért a változásokat, amint csak lehetséges, át kell vezetni az adatbázisba.

Egy adatbázis tetszőleges méretű és összetettségű lehet. A korábban említett, neveket és címeket tartalmazó lista például csak néhány száz, egyszerű szerkezetű rekordból állhat. Egy hatalmas könyvtár számítógépesített katalógusa viszont akár félmillió bejegyzést is tartalmazhat különböző — az első szerző vezetékneve, a mű tárgya, illetve címe szerinti — kategóriákba szervezve, az egyes kategóriákon belül ábécérendben. Ennél is nagyobb méretű és összetettségű az Internal Revenue Service (IRS) által karbantartott adatbázis, amely az amerikai adófizetők adóbevallásait tartalmazza. Ha feltételezzük, hogy százmillió adófizető van, akik közül mindenki átlagosan öt bevallást készít, amelyek mindegyike 400 karakternyi információt tartalmaz, akkor egy $100 \times 10^6 \times 400 \times 5$ karakter (bájt) mennyiségű információt tartalmazó adatbázist kapunk. Ha az IRS minden adófizető esetén az aktuálison kívül az utolsó három adóbevallást is megőrzi, akkor egy 8×10^{11} bájt (800 gigabájt) méretű adatbázist kapunk. Ezt a hatalmas mennyiségű információt úgy kell szervezni és kezelni, hogy a felhasználók szükség szerint kereshessék, lekérdezhessék és módosíthassák a benne szereplő adatokat. Nagy üzleti adatbázisra példa az Amazon.com, amely több mint 20 millió könyvről, CD-ről, videóról, DVD-ről, játékról, elektronikai eszközről, ruházati cikkről és egyéb termékről tartalmaz adatokat. Ez az adatbázis több mint 2 terabájtot foglal el (1 terabájt egyenlő 10^{12} bájttal), és 200 különböző számítógépen (adatbázisszerveren) tárolódik.

Naponta kb. 15 millió látogató éri el az Amazon.com-ot, és használja vásárlásaihoz az adatbázist. Az adatbázist folyamatosan aktualizálják, ahogy új könyveket és egyéb cikkeket vesznek nyilvántartásba, a raktáron lévő mennyiségeket pedig a vásárlások során frissítik. Nagyjából 100 ember felelős az Amazon adatbázisának karbantartásáért.

Az adatbázist létrehozhatjuk és karbantarthajuk kézzel vagy számítógép segítségével. Egy könyvtári katalógus például egy olyan adatbázis, amelyet kézzel hozunk létre és tartunk karban. Egy számítógéppel kezelt adatbázist vagy konkrétan erre a célra írt alkalmazásokkal hozunk létre és tartunk karban, vagy pedig egy adatbázis-kezelő rendszerrel. Ebben a jegyzetben csak számítógéppel kezelt adatbázisokkal foglalkozunk.

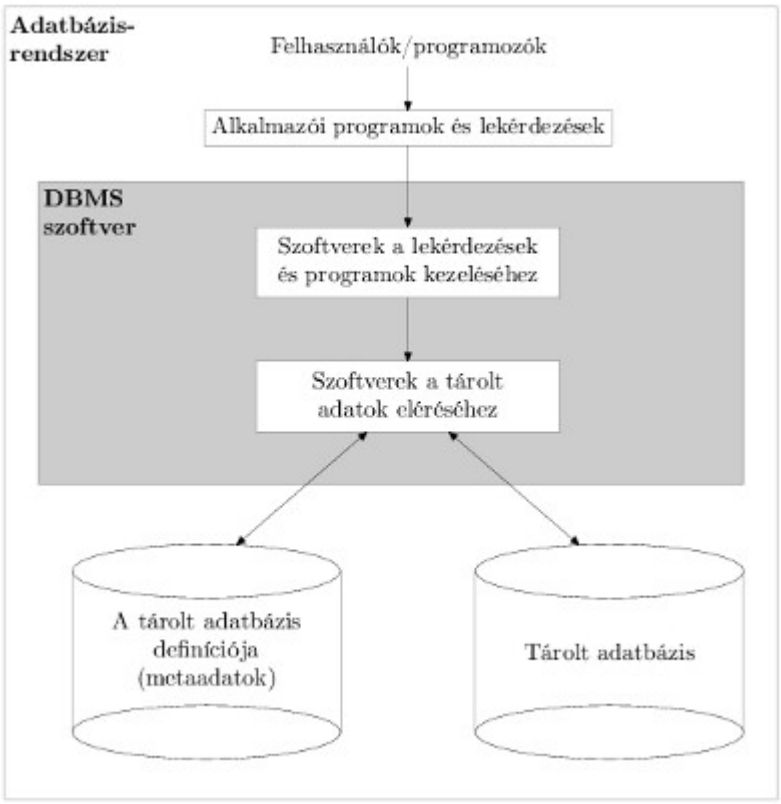
Az **adatbázis-kezelő rendszer (DBMS)** olyan programok gyűjteménye, amelyek lehetővé teszik a felhasználóknak, hogy létrehozzanak és karbantartsanak adatbázisokat. A DBMS egy *általános célú szoftverrendszer*, amely lehetővé teszi különböző felhasználók és alkalmazások számára az adatbázisok *definiálását*, *létrehozását*, *manipulálását* és *megosztását*. Az adatbázis **definiálása** az adatbázisban tárolandó adatokra jellemző adattípusok, adatszerkezetek és megszorítások megadását jelenti. Az adatbázis-definíció, azaz az adatbázist leíró információk szintén az adatbázisban tárolódnak adatbázis-katalógus vagy szótár formájában; ezeket **metaadatoknak** nevezzük. Az adatbázis **létrehozása** az a folyamat, amikor eltároljuk az adatokat valamilyen, a DBMS által vezérelt adathordozón. Az adatbázis **manipulálása** olyan funkciókat foglal magában, mint például az adatbázis lekérdezése konkrét adatok lekérése céljából, az adatbázis módosítása a minivilág változásának tükrözése céljából, vagy jelentések készítése az adatok alapján. Az adatbázis **megosztása** lehetővé teszi, hogy több felhasználó és program párhuzamosan hozzáférjen az adatbázishoz.

Az **alkalmazói programok** a DBMS felé küldött lekérdezésekkel vagy adatok iránti igényekkel férnek hozzá az adatbázishoz. A **lekérdezés** általában bizonyos adatok letöltését jelenti; míg a **tranzakció** bizonyos adatok olvasásával és bizonyos adatok írásával jár az adatbázisban.

A DBMS által biztosított egyéb fontos funkciók közé tartozik az adatbázis *védelme* és a hosszú időn át történő *karbantartása*. A **védelem** magában foglalja a hibás hardver- vagy szoftverműködés (összeomlás) elleni *rendszervédelmet*, valamint az illetéktelen vagy rosszindulatú hozzáférés elleni *biztonsági védelmet*. Egy tipikus nagy adatbázisnak többéves életciklusa lehet, a DBMS-nek tehát **karban** kell **tartania** az adatbázist, miközben hagyja, hogy a rendszer folyamatosan fejlődjön, ahogyan a követelmények idővel változnak.

Nem szükségszerűen kell általános célú DBMS szoftvert használnunk egy számítógéppel kezelt adatbázis megvalósításához. Írhatunk saját programokat is az adatbázis létrehozásához és karbantartásához, amellyel gyakorlatilag elkészítjük saját, *speciális célú* DBMS szoftverünket. Mindkét esetben — akár általános célú DBMS-t használunk, akár nem — rendszerint nagy mennyiségű összetett szoftvert kell telepítenünk. Valójában a legtöbb DBMS rendkívül összetett szoftverrendszer.

1.1. ábra - Az adatbázisrendszer leegyszerűsített sémája



Hogy teljessé tegyük a kezdeti definícióinkat, az adatbázist, a felhasználókat, a DBMS szoftvert, illetve a DBMS-t futtató és az adatbázist tároló hardvert együtt **adatbázisrendszernek** nevezzük. Az 1.1. ábra illusztrál néhányat az eddig tárgyalt fogalmak közül.

Egy példa

Tekintsünk egy egyszerű példát, amely a legtöbb olvasó számára ismerős lehet: egy EGYETEM adatbázist hallgatókra, tárgyakra és eredményekre vontakozó információk karbantartására egy egyetemi környezetben. Az 1.2. ábra bemutatja az adatbázis szerkezetét és néhány példa adatot egy ilyen adatbázishoz. Az adatbázist öt állományba (fájlba) szervezzük, melyek mindegyike azonos típusú **adatrekordokat** tárol.[1] A HALLGATÓ állomány az egyes diákokra, a TÁRGY állomány az egyes tárgyakra, a KURZUS állomány a tárgyak egyes kurzusaira vonatkozó adatokat tárolja, az INDEXSOR állomány azokat az eredményeket tárolja, amelyeket a hallgatók a különféle, általuk már teljesített kurzusokból elértek, míg az ELŐFELTÉTEL állomány az egyes tárgyak előfeltételeit tárolja.

1.2. ábra - Egy adatbázis, amely hallgatókról és tárgyakról tárol információkat.

HALLGATÓ

Név	Hallgatói_azonosító	Évfolyam	Szak
Kovács	17	1	PTI
Szabó	8	2	PTI

TÁRGY

Tárgynév	Tárgykód	Kredit	Tanszék
Bevezetés az informatikába	INDK201	5	IT
Adatszerkezetek	INDK421	5	IT
Diszkrét matematika	INDK101	4	AM
Adatbázisrendszerek	INDK501	4	IT

KURZUS

Kurzuskód	Tárgykód	Félév	Tanév	Oktató
85	INDK101	őszi	2009/10	Bácsó
92	INDK201	őszi	2009/10	Terdik
102	INDK421	tavaszi	2010/11	Kósa
112	INDK101	őszi	2010/11	Bácsó
119	INDK201	őszi	2010/11	Csernoch
135	INDK501	őszi	2010/11	Adamkó

INDEXSOR

Hallgatói_azonosító	Kurzuskód	Érdemjegy
17	112	4
17	119	3
8	85	5
8	92	5
8	102	4
8	135	5

ELŐFELTÉTEL

Tárgykód	Előfeltétel_kód
INDK501	INDK421
INDK501	INDK101
INDK421	INDK201

Egy adatbázis *definiálásához* az egyes rekordokban tárolandó, különböző típusú **adatelemek** definiálásával meg kell adnunk az egyes állományok rekordjainak a szerkezetét. Az 1.2. ábrán minden egyes HALLGATÓ rekord tartalmazza a hallgató nevét, hallgatói azonosítóját, évfolyamát (1, 2, és így tovább) és szakját (PTI a programtervező informatikus szakon, GI a gazdaságinformatikus szakon, stb.) reprezentáló adatokat; minden egyes TÁRGY rekord tartalmazza a tárgynevet, tárgykódot, a kreditszámot és a tanszéket (amelyik a tárgyat hirdeti) reprezentáló adatot; és így tovább. Egy rekordon belül meg kell adnunk egy **adattípust** is minden egyes adatelemhez. Például megadhatjuk, hogy a HALLGATÓ neve egy alfabetikus karaktersorozat, a HALLGATÓ hallgatói azonosítója egy egész szám, az INDEXSOR érdemjegye pedig szintén egy egész szám (az {1, 2, 3, 4, 5} halmazból). Használhatunk egy kódolási sémát is egy adatelem értékének a reprezentálására. Például az 1.2. ábrán 1-es érték reprezentálja egy elsőéves, 2-es egy másodéves, 3-as egy harmadéves, 4-es egy negyedéves, 5-ös pedig egy ötödéves

HALLGATÓ évfolyamát.

Az EGYETEM adatbázis *elkészítésekor* az egyes hallgatókat, tárgyakat, kurzusokat, indexsor-bejegyzéseket és előfeltételeket egy-egy rekordként tároljuk a megfelelő állományban. Megjegyzendő, hogy a különféle állományokban lévő rekordok kapcsolatban állhatnak egymással. Például a Kovácsra vonatkozó rekord a HALLGATÓ állományban két rekordhoz kapcsolódik az INDEXSOR állományban, amelyek megadják Kovács eredményeit két kurzusból. Hasonlóan, minden egyes rekord az ELŐFELTÉTEL állományban két tárgy rekordhoz kapcsolódik: az egyik a tárgyat reprezentálóhoz, a másik az előfeltételt reprezentálóhoz. A legtöbb közepes vagy nagy méretű adatbázis sokféle típusú rekordot tartalmaz, és *számos kapcsolat* létezik bennük a rekordok között.

Az adatbázis *manipulációja* a lekérdezést és a frissítést foglalja magában. Lekérdezésekre példák a következők:

- Készítsünk egy kimutatást 'Kovács' tanulmányairól — adjuk meg az összes, általa teljesített kurzus és a belőlük elért eredményeknek a listáját.
- Listázzuk ki azoknak a hallgatóknak a neveit, akik felvették az 'Adatbázisrendszerek' tárgy kurzusát a 2010/11-es tanév őszi félévében, és adjuk meg ezen hallgatók eredményeit is ebből a kurzusból.
- Listázzuk ki az 'Adatbázisrendszerek' tárgy előfeltételeit.

A frissítési példák a következőket foglalják magukba:

- Változtassuk 'Kovács' évfolyamát másodévesre.
- Hozzunk létre egy új kurzust az 'Adatbázisrendszerek' tárgyhoz erre a félévre.
- Írjunk be egy 5-ös érdemjegyet 'Kovács'-nak a legutóbbi félév 'Adatbázisrendszerek' kurzusára.

Ezeket az informális lekérdezéseket és frissítéseket a DBMS lekérdező nyelvében pontosan definiálni kell, mielőtt végrehajthatnánk őket.

Ezen a ponton célszerű az adatbázist egy nagyobb egység, egy szervezeten belüli információs rendszer részeként leírni. Egy vállalaton belül az Információtechnológia osztály tervezi meg és tartja karban az információs rendszert, amely különféle számítógépekből, tárolóeszközökből, alkalmazói szoftverekből és adatbázisokból áll. Egy létező adatbázishoz egy új alkalmazás tervezése, vagy egy új adatbázis tervezése a **követelménydefiníciónak és analízisnek** nevezett fázissal indul. Ezeket a követelményeket részletesen dokumentálják, majd átalakítják egy **koncepcionális tervvé**, amelyet számítógépes eszközökkel lehet reprezentálni és kezelni, ezáltal könnyen karbantartható és módosítható lesz, valamint könnyen át lehet alakítani egy adatbázis-implementációvá. A 3. fejezetben bemutatunk egy modellt, amely ezt a célt szolgálja, és amelyet egyed-kapcsolat modellnek nevezünk. A tervet ezután átalakítják **logikai tervvé**, amelyet egy kereskedelmi DBMS-ben implementált adatmodellel lehet kifejezni. Ebben a jegyzetben az 5. fejezettől kezdődően a relációsnak nevezett adatmodellre helyezzük a hangsúlyt. Jelenleg ez a legnépszerűbb megközelítése az adatbázisok tervezésének és implementálásának a (relációs) DBMS-ek körében. Az utolsó fázis a **fizikai tervezés**, amelynek során további specifikációkat adnak az adatbázis tárolására és elérésére vonatkozóan. Az adatbázistervet implementálják, feltöltik tényleges adatokkal, és folyamatosan karbantartják, hogy híven tükrözze a minivilág állapotát.

Az adatbázisszemlélet jellegzetességei

1.3. ábra - Az 1.2. ábra adatbázisából származó két nézet. (a) A LECKEKÖNYV nézet. (b) A TÁRGY_ELŐFELTÉTELEK nézet.

(a)

LECKEKÖNYV

Hallgató_név	Hallgató_leckekönyv				
	Tárgykód	Érdemjegy	Félév	Tanév	Kurzuskód
Kovács	INDK201	3	őszi	2010/11	119
	INDK101	4	őszi	2010/11	112
Szabó	INDK101	5	őszi	2009/10	85
	INDK201	5	őszi	2009/10	92
	INDK421	4	tavaszi	2010/11	102
	INDK501	5	őszi	2010/11	135

(b)

TÁRGY_ELŐFELTÉTELEK

Tárgynév	Tárgykód	Előfeltételek
Adatbázisrendszerek	INDK501	INDK421
		INDK101
Adatszerkezetek	INDK421	INDK201

A színpadi szereplők

Egy kis méretű, személyes adatbázis esetén (mint amilyen az . alfejezetben említett címjegyzék) tipikusan egy személy definiálja, hozza létre és kezeli az adatbázist, amely nincs megosztva. Nagy szervezetek esetén azonban sok ember érintett a több száz felhasználóval rendelkező nagy adatbázisok tervezésében, használatában és karbantartásában. Ebben a fejezetben bemutatjuk azokat az embereket, akiknek a feladatai közé tartozik egy nagy adatbázis mindennapi használata; őket hívjuk *színpadi szereplőknek*. Az . alfejezetben azokat a személyeket tekintjük át, akiket *háttérmunkásoknak* nevezhetünk: ők azok, akik azon dolgoznak, hogy fenntartsák az adatbázisrendszer környezetét, de nem foglalkoznak aktívan magával az adatbázissal.

Adatbázis-adminisztrátorok

Minden olyan szervezetnél, ahol sok ember használja ugyanazokat az erőforrásokat, szükség van egy vezető adminisztrátorra, aki felügyeli és kezeli azokat. Adatbázis környezetben az elsődleges erőforrás maga az adatbázis, a másodlagos erőforrás pedig a DBMS és a kapcsolódó szoftverek. Ezen erőforrások adminisztrálása az **adatbázis-adminisztrátor** (**DBA**) feladata. A DBA felelős az adabázishoz való hozzáférés engedélyezéséért, az adatbázis használatának koordinálásáért és felügyeletéért, valamint a szükséges szoftver- és hardvererőforrások beszerzéséért. A DBA kérhető számon olyan problémák esetén, mint amilyen a biztonsági rés vagy a lassú válaszidő. Nagy szervezeteknél a DBA-t egy stáb segíti, akik ellátják ezeket a feladatokat.

Adatbázis-tervezők

Az **adatbázis-tervezők** feladata az adatbázisban tárolandó adatok azonosítása, illetve ezen adatok reprezentálásához és tárolásához a megfelelő adatszerkezetek kiválasztása. Ezeket a műveleteket jórészt még azelőtt végrehajtják, mielőtt az adatbázist ténylegesen implementálnák és feltöltenék adatokkal. Az adatbázis-tervezők felelőssége, hogy beszéljenek az adatbázis összes leendő felhasználójával abból a célból, hogy megértsék az igényeiket, és hogy egy olyan tervet hozzanak létre, amely megfelel ezeknek az igényeknek. A tervezők gyakran a DBA-t segítő csapat tagjai, akik más feladatokat kapnak, miután az adatbázis tervezése befejeződött. Az adatbázis-tervezők rendszerint elbeszélgetnek az egyes potenciális felhasználói csoportokkal, majd olyan **nézeteket** fejlesztenek az adatbázishoz, amelyek megfelelnek ezen csoportok adat- és feldolgozási követelményeinek. Ezután minden nézetet elemeznek és *integrálnak* más felhasználói csoportok

nézeteivel. A végső adatbázistervnek képesnek kell lennie az összes felhasználói csoport igényeinek a támogatására.

Végfelhasználók

A **végfelhasználók** azok az emberek, akik a munkájuk során igénylik az adatbázishoz való hozzáférést lekérdezési, módosítási és jelentéskészítési célból; az adatbázis elsősorban értük létezik. A végfelhasználóknak több kategóriáját különböztetjük meg:

- Az **eseti végfelhasználók** csak időnként érik el az adatbázist, viszont minden alkalommal más-más információra lehet szükségük. Egy kifinomult adatbázis-lekérdező nyelvet használnak a kéréseik megfogalmazására; ők tipikusan közép- vagy felsővezetők, esetleg egyéb alkalomszerű böngészők.
- A **naiv** vagy **parametrikus végfelhasználók** számottevő részét teszik ki az adatbázis végfelhasználóinak. Munkájuk főleg az adatbázis állandó lekérdezése és módosítása körül forog, típuslekérdezéseket és -módosításokat használva, amelyeket **dobozolt tranzakcióknak** nevezünk, és amelyeket előzőleg gondosan leprogramoztak és teszteltek. Az ilyen felhasználók különféle feladatokat látnak el:
    - A banktisztviselők ellenőrzik a számlaegyenlegeket, és pénzkivételeket, -betéteket könyvelnek.
    - Légitársaságok, szállodák és autókölcsönzők ügyintézői ellenőrzik a kapott kérések teljesíthetőségét, és megteszik a helyfoglalást.
    - A szállítási cégek fogadóállomásain az ügyintézők csomagazonosítókat visznek be vonalkódok segítségével, valamint további leíró információkat billentyűzettel, így módosítva a megérkezett és a továbbszállítandó csomagok központi adatbázisát.
- A **tanult végfelhasználók** (szakemberek) magukban foglalják a mérnököket, tudósokat, üzleti elemzőket és másokat, akik alaposan megismerkednek a DBMS lehetőségeivel abból a célból, hogy saját alkalmazásokat fejlesszenek az összetett igényeik kielégítésére.
- A **független felhasználók** személyes adatbázisokat tartanak karban kész programcsomagok segítségével, amelyek egyszerűen használható menüs vagy grafikus interfészekkel rendelkeznek. Ilyen például egy olyan adócsomag felhasználója, amely személyes pénzügyi adatok sokaságát tárolja adózási célból.

Egy tipikus DBMS több eszközt biztosít az adatbázisok eléréséhez. A naiv végfelhasználóknak csak nagyon keveset kell ismerniük a DBMS által nyújtott szolgáltatásokból; nekik csupán a számukra tervezett és fejlesztett típustranzakciók felhasználói felületeit kell megérteniük. Az eseti felhasználók is csak az általuk újra és újra felhasznált eszközöket ismerik. A tanult felhasználók megpróbálják megismerni a DBMS legtöbb szolgáltatását, hogy kielégíthessék az összetett igényeiket. A független felhasználók általában meglehetősen jártasak lesznek egy bizonyos szoftvercsomag használatában.

Rendszerelemzők és alkalmazásprogramozók (szoftvermérnökök)

A **rendszerelemzők** határozzák meg a végfelhasználók, különösen a naiv felhasználók igényeit, majd rögzítik az ezen igényeknek megfelelő dobozolt tranzakciók specifikációit. Az **alkalmazásprogramozók** a megadott specifikációk alapján implementálják ezeket a tranzakciókat mint programokat, amelyeket azután tesztelnek, nyomkövetnek, dokumentálnak és karbantartanak. Az elemzőknek és a programozóknak — akiket közös néven **szoftverfejlesztőknek** vagy **szoftvermérnököknek** hívnak — a DBMS által nyújtott szolgáltatások teljes tárházával tisztában kell lenniük ahhoz, hogy feladatukat ellássák.

A háttérmunkások

Egy adatbázis tervezői, felhasználói és adminisztrátorai mellett mások a DBMS *szoftver- és rendszerkörnyezetét* tervezik, fejlesztik és működtetik. Ezek a személyek általában nem foglalkoznak magával az adatbázissal. Őket nevezzük *háttérmunkásoknak*, akik a következő kategóriákba sorolhatók:

- A **DBMS rendszertervezők és programozók** tervezik és implementálják a DBMS moduljait és interfészeit mint szoftvercsomagokat. A DBMS egy nagyon összetett szoftverrendszer, amely sok komponensből, **modulból** áll. Ide tartoznak a katalógust megvalósító, a lekérdezőnyelvet feldolgozó, az interfészt feldolgozó, az adatok elérését és pufferelését biztosító, a konkurenciakezelő vagy az adatvisszaállítást és biztonságot kezelő modulok. A DBMS-nek felületet kell biztosítania más rendszerszoftverekhez, mint például az operációs rendszerhez vagy a különböző programozási nyelvek fordítóprogramjaihoz.
- Az **eszközfejlesztők eszközöket** terveznek és implementálnak. Az eszközök olyan szoftvercsomagok, amelyek megkönnyítik az adatbázis-modellezést és -tervezést, az adatbázisrendszer tervezését és a teljesítmény növelését. Olyan opcionális csomagokról van szó, amelyeket gyakran külön kell megvásárolni. Ide tartoznak az adatbázis-tervezésre, teljesítménymonitorozásra, természetes nyelvi vagy grafikus felületek kezelésére, prototípus-kezelésre, szimulációra és tesztadat-generálásra szolgáló csomagok. Sok esetben független szoftvergyártók fejlesztik és árulják ezeket az eszközöket.
- Az **operátorok és a karbantartó személyzet** (rendszeradminisztrációs személyzet) felelős az adatbázisrendszer hardver- és szoftverkörnyezetének tényleges futtatásáért és karbantartásáért.

Bár a felsorolt kategóriákba tartozó háttérmunkások tevékenyen közreműködnek abban, hogy az adatbázisrendszer elérhető legyen a végfelhasználók számára, saját céljaikra általában nem használják az adatbázist.

A DBMS megközelítés használatának előnyei

A redundancia kezelése

1.4. ábra - A Hallgató_név és a Tárgykód redundáns tárolása az INDEXSOR-ban. (a) Konzisztens adatok. (b) Inkonzisztens rekord.

(a)

INDEXSOR

Hallgatói_azonosító	Hallgató_név	Kurzuskód	Tárgykód	Érdemjegy
17	Kovács	112	INDK201	4
17	Kovács	119	INDK101	3
8	Szabó	85	INDK101	5
8	Szabó	92	INDK201	5
8	Szabó	102	INDK421	4
8	Szabó	135	INDK301	5

(b)

INDEXSOR

Hallgatói_azonosító	Hallgató_név	Kurzuskód	Tárgykód	Érdemjegy
17	Szabó	112	INDK101	4

A jogosulatlan hozzáférés korlátozása

Perzisztens tárolóhely biztosítása a program objektumainak számára

Társzerkezetek biztosítása a hatékony lekérdezésfeldolgozáshoz

Mentési és visszaállítási lehetőség biztosítása

Több felhasználói felület biztosítása

Az adatok közötti összetett kapcsolatok reprezentálása

Az integritási megszorítások kikényszerítése

Következtetések és tevékenységek szabályok használatával

Az adatbázis megközelítés használatának további következményei

Ebben az alfejezetben az adatbázis megközelítés használatának további olyan következményeit tárgyaljuk, amelyekből a legtöbb szervezet profitálhat.

Szabványok kikényszerítésének a lehetősége.

Rövidebb alkalmazásfejlesztési idő.

Rugalmasság.

Naprakész információk rendelkezésre állása.

Gazdaságos működés biztosítása a skálázhatósággal.

[1] Az *állomány* (*fájl*) fogalmát itt informálisan használjuk. Koncepcionális szinten egy *állomány* nem más, mint rekordoknak egy *gyűjteménye* (*kollekciója*), amely lehet rendezett vagy rendezetlen.

2. fejezet - Az adatbázisrendszer alapfogalmai és architektúrája

Tartalom

Adatmodellek, sémák és példányok

Az adatbázis megközelítés egyik alapvető jellemzője, hogy egy bizonyos fokú adatabsztrakciót biztosít. Az **adatabsztrakció** általánosan az adatok szervezésére és tárolására vonatkozó részletek elhagyását, illetve a leglényegesebb tulajdonságok kiemelését jelenti az adatok jobb megértése érdekében. Az adatbázis megközelítés egyik fő jellemzője, hogy támogatja az adatabsztrakciót, s így a különböző felhasználók az általuk preferált részletességi szinten láthatják az adatokat. Ezt az absztrakciót az adatmodell biztosítja. Az **adatmodell** olyan eszközök összessége, amelyek segítségével leírható egy adatbázis szerkezete. Az *adatbázis szerkezetén* az adattípusokat, az adatok közötti kapcsolatokat és a rájuk vonatkozó megszorításokat értjük. A legtöbb adatmodell az adatbázis lekérdezését és módosítását leíró **alapműveleteket** is tartalmaz.

Manapság egyre gyakoribb, hogy az adatmodell az alapműveleteken kívül olyan eszközöket is tartalmaz, amelyekkel az adatbázis alkalmazások **dinamikus aspektusa** vagy **viselkedése** is leírható. Ez lehetővé teszi, hogy az adatbázis tervezője megadhassa az adatbázis objektumain végezhető érvényes felhasználói műveleteket. Ilyen műveletre példa az `ÁTLAGSZÁMÍTÁS`, amely `HALLGATÓ` objektumokra alkalmazható. Másrészről viszont azokat az általános műveleteket, amelyek egy tetszőleges objektum beszúrására, törlésére, módosítására vagy lekérdezésére szolgálnak, gyakran *az adatmodell alapműveletei* tartalmazzák. A viselkedés leírására szolgáló eszközök alapvetőek az objektumorientált adatmodellekben, de már a hagyományosabb adatmodellekben is kezdenek megjelenni. Az objektum-relációs modellek például az alap relációs modellt egészítik ki többek között ilyen eszközökkel. A relációs adatmodellben is tervezik, hogy a relációkhoz viselkedést lehessen hozzárendelni perzisztens tárolt modulok, népszerű nevén tárolt eljárások formájában.

Az adatmodellek csoportosítása

Számos adatmodellt terveztek, amelyeket az adatbázis szerkezetének leírására általuk használt eszközök típusai szerint csoportosíthatunk. A **magas szintű** vagy **koncepcionális adatmodellek** olyan eszközöket nyújtanak, amelyek közel állnak ahhoz, ahogyan a legtöbb felhasználó látja az adatokat, míg az **alacsony szintű** vagy **fizikai adatmodellek** által nyújtott eszközök részletesen leírják, hogyan tárolódnak az adatok a számítógépen. Az alacsony szintű adatmodellek által biztosított eszközök általában számítógépes szakemberek számára készültek, nem a tipikus végfelhasználók számára. E két véglet között helyezkednek el a **reprezentációs** (vagy **implementációs**) **adatmodellek**. Ezek a modellek olyan eszközöket nyújtanak, amelyeket a végfelhasználók is megérthetnek, de nem állnak messze attól sem, ahogyan az adatok a számítógépen szerveződnek. A reprezentációs adatmodellek elrejtik az adattárolás bizonyos részleteit, ugyanakkor közvetlenül implementálhatók egy számítógépes rendszeren.

A koncepcionális adatmodellek olyan fogalmakat használnak, mint például az egyed, a tulajdonság és a kapcsolat. Az **egyed** egy olyan valós világbeli objektumot vagy fogalmat reprezentál, amely szerepel az adatbázisban, mint például egy alkalmazott vagy egy projekt. A **tulajdonság** vagy **attribútum** az egyedet leíró, a modell szempontjából érdekes jellemző, mint például az alkalmazott neve vagy fizetése. Egy két vagy több egyed közötti **kapcsolat** az egyedek között fennálló valamilyen viszonyt jelöl. Ilyen például a „dolgozik rajta” kapcsolat egy alkalmazott és egy projekt között. A későbbiekben bemutatásra kerülő egyed-kapcsolat (Entity-Relationship, ER) modell egy népszerű magas szintű koncepcionális adatmodell. Az ER modell kiegészíthető további absztrakciós eszközökkel, mint amilyen a generalizáció, a specializáció és a kategória.

A reprezentációs vagy implementációs adatmodellek azok, amelyeket a leggyakrabban használnak a hagyományos kereskedelmi DBMS-ekben. Ilyen például a széles körben használt relációs adatmodell, valamint a korai adatmodellek — a **hálós** és a **hierarchikus** modellek —, amelyeket a múltban széles körben használtak. Hamarosan részletesen bemutatjuk a relációs adatmodellt, a műveleteit és nyelveit. A reprezentációs adatmodellek az adatokat rekord szerkezetekben tárolják,

ezért néha **rekord alapú adatmodelleknek** is nevezik őket.

Az **objektum adatmodell csoport** (object data model group, ODMG) olyan magasabb szintű implementációs adatmodellek egy új családjának tekinthető, amelyek közelebb állnak a koncepcionális adatmodellekhez. A későbbiekben bemutatjuk az objektum-adatbázisok és az ODMG által javasolt szabvány általános jellemzőit. Az objektum adatmodelleket gyakran használják magas szintű koncepcionális modellekként is, elsősorban a szoftvertervezés területén.

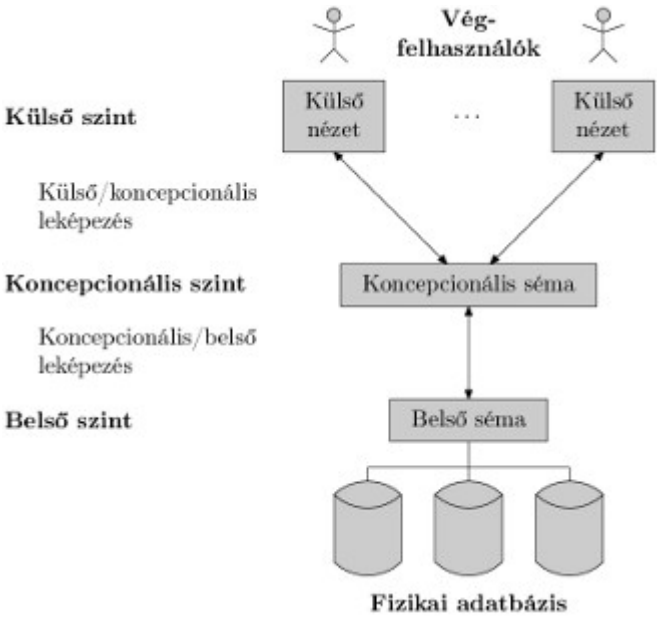
Sémák, példányok és adatbázis-állapot

A háromséma architektúra és az adatfüggetlenség

Az adatbázisszemléletnek a . alfejezetben felsorolt négy fontos jellegzetessége közül három: (1) a programok és adatok szétválasztása (program-adat és program-művelet függetlenség), (2) több felhasználói nézet támogatása, és (3) egy katalógus használata az adatbázis leírásának (a sémának) a tárolására. Ebben az alfejezetben közelebbről megadjuk az adatbázisrendszerek egy architektúráját, az ún. **háromséma architektúrát**, amelyenek a célja, hogy segítsen megérteni és megjeleníteni ezeket a jellegzetességeket. Ezután a továbbiakban az adatfüggetlenség fogalmát tárgyaljuk.

A háromséma architektúra

2.1. ábra - A háromséma architektúra



A 2.1. ábrán látható háromséma architektúra célja, hogy elkülönítse a felhasználói alkalmazásokat és a fizikai adatbázist. Ebben az architektúrában a következő három szinten definiálhatunk sémákat:

1. A **belső szintnek** van egy **belső sémája**, amely az adatbázis fizikai tárolási szerkezetét írja le. A belső séma egy fizikai adatmodellt használ, és leírja az adattárolás összes részletét, valamint a hozzáférési utakat az adatbázishoz.
2. A **koncepcionális szintnek** van egy **koncepcionális sémája**, amely a teljes adatbázis szerkezetét írja le a felhasználók összességének. A koncepcionális séma elrejti a fizikai tárolási szerkezetek részleteit, csak az egyedek, adattípusok, kapcsolatok, felhasználói műveletek és megszorítások leírására korlátozódik. Általában egy reprezentációs adatmodellt szoktunk használni a koncepcionális séma leírására, amikor megvalósítunk egy adatbázist. Ez az *implementációs koncepcionális séma* gyakran egy *koncepcionális*

sématerven alapul egy magas szintű adatmodellben.

3. A **külső (nézet) szint** számos **külső sémát (felhasználói nézetet)** foglal magában. Mindegyik külső séma az adatbázisnak egy olyan részét írja le, amely iránt egy bizonyos felhasználói csoport érdeklődik, és elrejti ez elől a felhasználói csoport elől az adatbázis többi részét. Az előző esethez hasonlóan mindegyik külső sémát jellemzően egy reprezentációs adatmodell felhasználásával implementáljuk, esetleg egy magas szintű adatmodellbeli külső sémára alapozva.

A háromséma architektúra egy kényelmes eszköz, amellyel a felhasználó maga elé képzelheti a sémaszinteket egy adatbázisrendszerben. A legtöbb adatbázis-kezelő rendszer nem választja szét teljesen és határozottan a három szintet, de bizonyos mértékben támogatja a háromséma architektúrát. Vannak adatbázis-kezelő rendszerek, amelyek a fizikai szintű részleteket beleértik a koncepcionális sémába. A háromszintű ANSI architektúrának fontos szerepe van az adatbázis technológia fejlesztésében, mert világosan szétválasztja a felhasználó külső szintjét, a rendszer koncepcionális szintjét és a belső tárolási szintet egy adatbázis tervezésekor. Manapság is jól alkalmazható adatbázis-kezelő rendszerek tervezésében. A legtöbb adatbázis-kezelő rendszer, amely támogatja a felhasználói nézeteket, a külső sémákat ugyanabban az adatmodellben adja meg, amely leírja a koncepcionális szintű információkat (például egy olyan relációs adatbázis-kezelő rendszer, mint az Oracle, az SQL-t használja erre). Néhány adatbázis-kezelő rendszer megengedi különböző adatmodellek használatát a koncepcionális és a külső szinteken. Példa erre az Universal Data Base (UDB), az IBM adatbázis-kezelő rendszere, amely relációs adatmodellt használ a koncepcionális séma leírására, egy külső séma leírásához viszont objektumorientált modellt is lehet használni benne.

Vegyük észre, hogy a három séma csak a *leírása* az adatoknak; a tárolt adatok, amelyek *valójában* léteznek, a fizikai szinten találhatók. Egy háromséma architektúrára alapozott adatbázis-kezelő rendszerben minden felhasználói csoport csak a saját külső sémájára tud hivatkozni. Ezért az adatbázis-kezelő rendszernek egy külső sémára vonatkozó kérést át kell alakítania egy olyan kéréssé, amely a koncepcionális sémán értelmezhető, majd aztán egy belső sémára vonatkozó kéréssé a fizikai adatbázis feldolgozásához. Ha egy kérés egy adatbázis-lekérdezés, a fizikai adatbázisból kinyert adatokat vissza kell alakítani úgy, hogy az illeszkedjen a felhasználó külső nézetéhez. A kérések és eredmények egyes szintek közötti átalakításának a folyamatát **leképezéseknek** hívjuk. Ezek a leképezések időigényesek lehetnek, ezért néhány adatbázis-kezelő rendszer — különösen azok, amelyeket kis adatbázisok támogatására terveztek — nem támogatja a külső nézeteket. Mindazonáltal még az ilyen rendszerekben is szükség van bizonyos mennyiségű leképezésre a kérések átalakításához a koncepcionális és a belső szintek között.

Az adatfüggetlenség

A háromséma architektúra az **adatfüggetlenség** fogalmának további kifejtésére is felhasználható, amely definiálható úgy is, mint a séma megváltoztatásának a képessége egy adatbázisrendszer valamely szintjén anélkül, hogy meg kellene változtatni a sémát a következő magasabb szinten. Az adatfüggetlenségnek két típusát definiálhatjuk:

1. A **logikai adatfüggetlenség** a koncepcionális séma megváltoztatásának a képessége anélkül, hogy meg kellene változtatni a külső sémákat vagy az alkalmazói programokat. Megváltoztathatjuk a koncepcionális sémát az adatbázis kiterjesztésével (egy rekordtípus vagy egy sémaelem hozzáadásával), a megszorítások módosításával, vagy az adatbázis csökkentésével (egy rekordtípus vagy egy sémaelem eltávolításával). Az utóbbi eset azokra a külső sémákra, amelyek csak a megmaradó adatokra hivatkoznak, nem lehet hatással. Az 1.3. (a) ábrán látható külső séma például nem változhat meg, ha a 1.2. ábrán látható INDEXSOR állomány (vagy rekordtípus) az 1.4. ábrán láthatóra módosul. Egy adatbázis-kezelő rendszerben, amely támogatja a logikai adatfüggetlenséget, csak a nézetdefiníciókat és a leképezéseket kell megváltoztatni. Miután a koncepcionális séma keresztülment a

logikai újraszervezésen, az alkalmazói programoknak, amelyek a külső sémaelemekre hivatkoztak, ugyanúgy kell működniük, mint azelőtt. A megszorítások módosításai anélkül alkalmazhatók a koncepcionális sémára, hogy hatással lennének a külső sémákra vagy az alkalmazói programokra.

2. A **fizikai adatfüggetlenség** a belső séma megváltoztatásának a képessége anélkül, hogy meg kellene változtatni a koncepcionális sémát. Ennélfogva a külső sémákat sincs szükség megváltoztatni. A belső sémát érintő változások azért válhatnak szükségessé, mert néhány fizikai állományt újraszervezünk — például további hozzáférési szerkezetek létrehozásával — a lekérdezések vagy a módosítási műveletek hatékonyságának növelése érdekében. Ha ezek után ugyanazok az adatok maradnak az adatbázisban, akkor elvileg nem kell megváltoztatnunk a koncepcionális sémát. Ha például a sebesség növelése érdekében biztosítunk egy elérési utat a kurzus rekordok (1.2. ábra) tanév és félév szerinti lekérdezéséhez, elvileg nem kell megváltoztatnunk egy olyan lekérdezést, mint például hogy *listázzuk ki a 2009/10-es tanév őszi félévében meghirdetett kurzusokat*, bár a lekérdezést a DBMS hatékonyabban fogja végrehajtani az új elérési út használatával.

A fizikai adatfüggetlenség a legtöbb adatbázisban és állománykörnyezetben jelen van; ekkor az adatok pontos elhelyezkedése a lemezen, a háttértáron való tárolás miként je, a rekordok elhelyezése, tömörítése, szétdarabolása és összeillesztése stb. rejtve vannak a felhasználók elől. Az alkalmazásoknak nem szükséges tudniuk ezekről a részletekről. Másrészről a logikai adatfüggetlenséget nagyon nehéz elérni, mivel az szerkezeti és megszorításbeli változásokat is megenged anélkül, hogy azok az alkalmazói programokra hatással lennének, és ez sokkal szigorúbb követelmény.

Ha többszintű DBMS-sel rendelkezünk, a katalógusát ki kell egészíteni olyan információkkal, amelyek azt mondják meg, hogy hogyan kell leképezni a kéréseket és az adatokat a különböző szintek között. A DBMS további szoftvereket használ ezeknek a leképezéseknek a megvalósítására, amelyek a katalógusban szereplő leképezési információkat veszik alapul. Az adatfüggetlenség ott jelenik meg, hogy amikor a séma egy adott szinten módosul, akkor a következő, magasabb szinten lévő séma változatlan marad; csak a két szint közötti *leképezés* változik. Így az alkalmazói programokat, amelyek a magasabb szintű sémára hivatkoznak, nem kell módosítani.

A háromséma architektúra megkönnyíti az igazi adatfüggetlenség elérését mind fizikai, mind logikai szinten. A leképezések két szintje azonban plusz munkát követel meg egy lekérdezés vagy program lefordítása vagy végrehajtása közben, és ez csökkenti a DBMS hatékonyságát. Emiatt csak kevés DBMS implementálja a teljes háromséma architektúrát.

Az adatbázisrendszer környezete

Az adatbázis-kezelő rendszerek osztályozása

3. fejezet - Adatmodellezés az ER modell segítségével

Tartalom

A koncepcionális modellezés nagyon fontos fázisa egy sikeres adatbázis alkalmazás tervezésének. Az **adatbázis alkalmazás** kifejezés általánosságban egy konkrét adatbázisra és a hozzá kapcsolódó programokra utal, amelyek az adatbázis-lekérdezéseket és -módosításokat valósítják meg. Egy BANK adatbázis alkalmazás például, amely az ügyfelek számláit tartja nyilván, olyan programokat tartalmazhat, amely az ügyfelek pénzmozgásainak megfelelő adatbázis-módosításokat implementálja. Ezek a programok felhasználóbarát grafikus felhasználói felületekkel (GUI-kkal) rendelkeznek, amelyek űrlapokat és menüket biztosítanak az alkalmazás végfelhasználói (jelen esetben a banki ügyintézők) számára. Ezért az adatbázis alkalmazás részét képezi ezen alkalmazói programok tervezése, implementációja és tesztelése is. Az **alkalmazói programok** tervezését és tesztelését hagyományosan inkább a szoftvertervezés területéhez tartozónak tekintik, mintsem az adatbázis-tervezéshez kapcsolódónak. Mivel azonban az adatbázis-tervezési módszertanok egyre több, az adatbázis-objektumokon végzendő műveletekre vonatkozó eszközt tartalmaznak, a szoftvertervezési módszertanok pedig egyre részletesebben meghatározzák a programok által használt és elért adatbázis szerkezetét, nyilvánvaló, hogy ezek a tevékenységek szorosan összetartoznak.

Ebben a fejezetben a hagyományos megközelítést követjük, azaz az adatbázis-tervezés során az adatbázis szerkezetére és megszorításaira koncentrálunk. Ismertetjük az **egyed-kapcsolat** (**ER**) **modell** fogalmait, amely egy népszerű magas szintű koncepcionális adatmodell. Ezt a modellt és különböző változatait gyakran használják az adatbázis alkalmazások koncepcionális tervezésénél, és sok adatbázis-tervező eszköz használja a fogalmait. Bemutatjuk az ER modell alapvető adatszerkezeti fogalmait és megszorításait, és leírjuk ezek használatát az adatbázis alkalmazások koncepcionális sémáinak tervezése során. Ismertetjük az ER modellhez kötődő grafikus jelölésrendszert is, amelyet **ER diagramnak** nevezünk.

Ez a fejezet a következőképpen épül föl: A . alfejezet a magas szintű koncepcionális adatmodelleknek az adatbázis-tervezésben játszott szerepét tárgyalja. A . alfejezetben egy példa adatbázis alkalmazás követelményeit adjuk meg, hogy illusztráljuk az ER modell fogalmainak a használatát. Ezt a példa adatbázist használjuk a következő fejezetekben is. A . alfejezetben definiáljuk az egyed és az attribútum fogalmát, és fokozatosan bevezetjük az ER séma grafikus megjelenítésének technikáját. A . alfejezetben a bináris kapcsolatokról, azok szerepéről és a strukturális megszorításokról ejtünk szót. A . alfejezet a gyenge egyedtípusokról szól. A . alfejezet megmutatja, hogyan lehet egy sématervet úgy finomítani, hogy kapcsolatokat is tartalmazzon. A . alfejezet az ER diagramok jelöléseit ismerteti, összefoglalja a sématervezés során felmerülő kérdéseket, és leírja, hogy hogyan kell az adatbázissémában található konstrukciók neveit

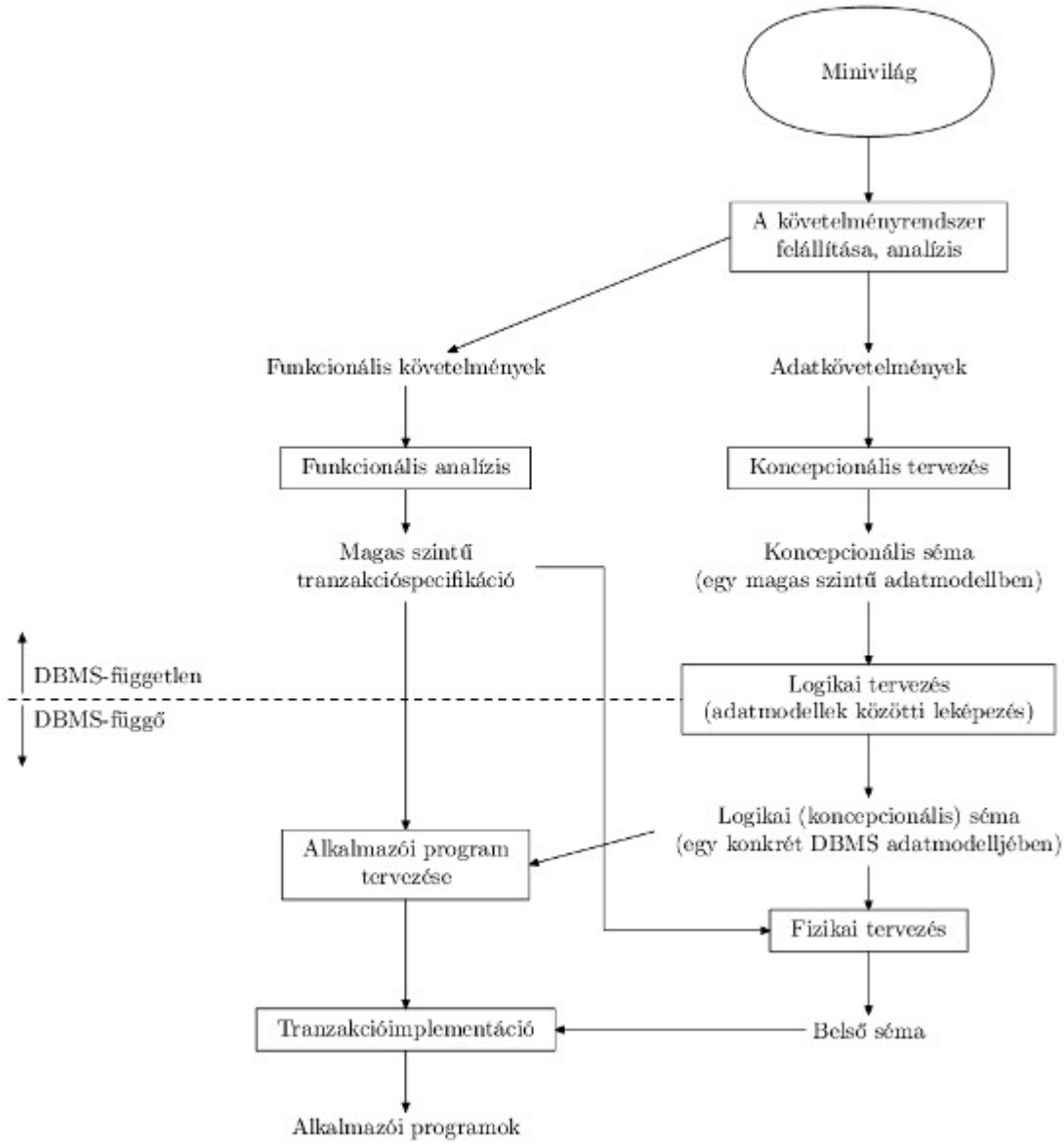
megválasztani. A . alfejezet a kapcsolatok összetettebb típusait tárgyalja. A . alfejezet összefoglalja a fejezetet.

A . alfejezet anyaga elhagyható egy bevezető kurzusból. Ha az olvasó az adatmodellezési fogalmakban és a koncepcionális adatbázis-tervezésben alaposabb jártasságot szeretne szerezni, akkor a . alfejezet után a 4. fejezettel folytathatja, amelyben az ER modell olyan kiterjesztéseit ismertetjük, melyek a kibővített ER (EER) modellhez vezetnek. Az EER modell olyan fogalmakat tartalmaz, mint a specializáció, a generalizáció, az öröklődés és az unió típusok (kategóriák).

Magas szintű koncepcionális adatmodellek használata az adatbázis-tervezésben

A 3.1. ábra az adatbázis-tervezési folyamat egy leegyszerűsített leírását mutatja. Az első lépés **a követelményrendszer felállítása és analízise**. Ebben a lépésben az adatbázis-tervezők elbeszélgetnek az adatbázis leendő felhasználóival, hogy megértsék és dokumentálják az **adatkövetelményeiket**. Ennek a lépésnek az eredménye a felhasználók követelményeinek írásban rögzített, lényegretörő listája. Ezeket a követelményeket olyan részletes és teljes formában kell specifikálni, amennyire csak lehetséges. Az adatkövetelmények megadásával párhuzamosan hasznos, ha az alkalmazás ismert **funkcionális követelményeit** is megadjuk. Ezek azokból a felhasználó által definiált **műveletekből** (vagy **tranzakciókból**) állnak, amelyeket az adatbázisra fogunk alkalmazni, beleértve a lekérdezéseket és a módosításokat is. A szoftvertervezésben leggyakrabban *adatfolyam-diagramokat*, *szekvenciadiagramokat*, *forgatókönyveket* és más technikákat alkalmazunk a funkcionális követelmények leírására. Egyik felsorolt technikát sem tárgyaljuk itt; ezeket részletesen a szoftvertervezésről szóló szakirodalmak írják le.

3.1. ábra - Egy egyszerűsített ábra az adatbázis-tervezés főbb fázisainak illusztrálására.



Amint az összes követelményt összegyűjtöttük és elemeztük, a következő lépés az adatbázis **koncepcionális sémájának** létrehozása egy magas szintű koncepcionális adatmodell felhasználásával. Ezt a lépést **koncepcionális tervezésnek** nevezzük. A koncepcionális séma a felhasználók adatkövetelményeinek egy tömör leírása, és az egyedtípusok, a kapcsolatok és a megszorítások részletes leírását tartalmazza; ezeket a magas szintű adatmodell eszközeivel fejezzük ki. Mivel ezek az eszközök nem tartalmaznak megvalósítási részleteket, a laikus felhasználók számára rendszerint könnyebben érthetők, így a velük folytatott kommunikáció során jól használhatók. A magas szintű koncepcionális séma hivatkozásként is használható annak érdekében, hogy az összes felhasználó adatkövetelményei egységesen jelenjenek meg, és hogy ezek a követelmények ne mondjanak ellent egymásnak. Ez a megközelítés lehetővé teszi az adatbázis-tervezők számára, hogy az adatok tulajdonságait a tárolás részleteinek figyelembevétele nélkül specifikálják. Ennek következtében könnyebben tudnak jó koncepcionális adatbázistervet készíteni.

A koncepcionális séma tervezése közben vagy után az adatmodell alapvető műveleteit használhatjuk azoknak a magas szintű felhasználói műveleteknek a megadására, amelyeket a funkcionális elemzés során azonosítottunk. Ez azt a célt is szolgálja, hogy meggyőződjünk arról, hogy a koncepcionális séma minden funkcionális követelménynek megfelel. Amennyiben bizonyos funkcionális követelményeket nem tudunk leírni a kezdeti séma segítségével, akkor módosíthatjuk a koncepcionális sémát.

Az adatbázis-tervezés következő lépése az adatbázis tényleges megvalósítása egy kereskedelmi DBMS felhasználásával. A legújabb kereskedelmi DBMS-ek egy implementációs adatmodellt — mint amilyen a relációs vagy az objektum-relációs adatmodell — használnak, a koncepcionális sémát tehát a magas szintű adatmodellről az implementációs adatmodellre alakítjuk át. Ezt a lépést **logikai tervezésnek** vagy **az adatmodell leképezésének** nevezzük; ennek eredménye egy, a DBMS implementációs adatmodelljében leírt adatbázisséma.

Az utolsó lépés a **fizikai tervezés** fázisa, amelynek során megadjuk az adatbázis állományainak belső tárolási szerkezetét, indexeit, elérési útjait és állományszervezési módjait. Ezekkel a tevékenységekkel párhuzamosan tervezzük meg és implementáljuk az alkalmazói programokat mint adatbázis tranzakciókat a magas szintű tranzakcióspecifikációnak megfelelően.

Ebben a fejezetben csak a koncepcionális sématervezéshez legszükségesebb ER modellbeli fogalmakat ismertetjük. További modellezési fogalmakat vezetünk be a 4. fejezetben, ahol az EER modellről tárgyalunk.

Egy példa adatbázis alkalmazás

Ebben az alfejezetben a VÁLLALAT nevű példa adatbázis alkalmazást mutatjuk be, amely az alapvető ER modellbeli fogalmakat és azoknak a sématervezés során történő felhasználását illusztrálja majd. Most felsoroljuk az adatbázis adatkövetelményeit, a későbbiek folyamán pedig lépésről lépésre elkészítjük a koncepcionális sémáját, párhuzamosan az ER modell modellezési fogalmainak a bevezetésével. A VÁLLALAT adatbázis egy vállalat dolgozóit, osztályait és projektjeit tartja nyilván. Tegyük fel, hogy a követelménygyűjtési és elemzési fázis után az adatbázis-tervezők a *minivilág* (a vállalat azon része, amelyet az adatbázisban reprezentálni szeretnénk) alábbi leírását adják meg:

- A vállalat osztályokból áll. Minden osztálynak egyedi neve és egyedi azonosítószáma van, valamint tartozik hozzá egy konkrét dolgozó, aki az osztályt vezeti. Nyilvántartjuk azt a kezdő dátumot, amikor ez a dolgozó vezetni kezdte az osztályt. Egy osztály számos helyszínnel rendelkezhet.
- Egy osztály számos projektet irányít, amelyek mindegyikének van egy egyedi neve, egy egyedi száma és egyetlen helyszíne.
- Tároljuk az egyes dolgozók nevét, személyi számát, lakcímét, fizetését, nemét és születési dátumát. Egy dolgozó egy osztályhoz van hozzárendelve, azonban több projekten is dolgozhat, amelyeket nem feltétleül ugyanaz az osztály irányít. Nyilvántartjuk, hogy egy dolgozó az egyes projekteken hetente hány órát dolgozik. Nyilvántartjuk az egyes dolgozók közvetlen főnökét is.
- Nyilván szeretnénk tartani az egyes dolgozók hozzátartozóit is biztosítási célokból. Tároljuk az egyes hozzátartozók keresztnevét, nemét, születési dátumát, valamint azt, hogy milyen kapcsolatban áll a dolgozóval.

A 3.2. ábra mutatja, hogy hogyan ábrázolhatjuk ennek az adatbázis alkalmazásnak a sémáját azzal a grafikus jelöléssel, amelyet **ER diagramnak** nevezünk. Ezt az ábrát fokozatosan fogjuk majd elmagyarázni, ahogyan az ER modell fogalmait ismertetjük. Azt a folyamatot, amelynek során a fenti követelményekből származtatjuk ezt a sémát, az ER modell fogalmainak bevezetésével párhuzamosan fogjuk lépésről lépésre bemutatni, miközben az ER diagramok jelöléseit is elmagyarázzuk.

3.2. ábra - Egy ER séma diagram a VÁLLALAT adatbázishoz. Az ábra grafikus jelöléseit ebben a fejezetben fokozatosan ismertetjük.

Egyedtípusok, egyedhalmazok, attribútumok és kulcsok

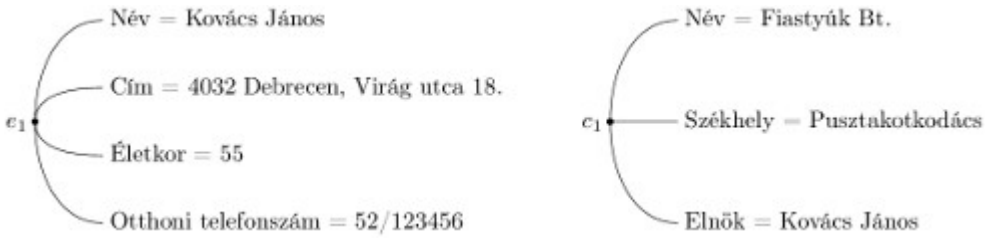
Az ER modell az adatokat mint *egyedeket*, *kapcsolatokat* és *attribútumokat* írja le. A . alfejezetben ismertetjük az egyedek és attribútumaik fogalmát. Az egyedtípusokat és kulcs attribútumokat a . alfejezetben tágyaljuk. Aztán a . alfejezetben megadjuk az egyedtípusok egy lehetséges koncepcionális tervét a VÁLLALAT adatbázishoz. A kapcsolatokat a . alfejezetben tárgyaljuk.

Egyedek és attribútumok

Egyedek és attribútumaik. Az ER modell által kezelt alapvető objektum az **egyed**, amely a valós világnak egy olyan *darabja*, amely önálló léttel bír. Az egyed lehet fizikai szinten létező objektum (például személy, autó, ház, dolgozó) vagy lehet fogalmi szinten létező objektum (például vállalat, foglalkozás vagy egy egyetemi tantárgy). Minden egyednek vannak **attribútumai** — az őt leíró tulajdonságok. Például egy dolgozó egyedet a dolgozó nevével, életkorával, címével, fizetésével és foglalkozásával lehet leírni. Egy konkrét egyed minden egyes attribútumához tartozik egy érték. Az egyedeket leíró attribútumértékek fogják alkotni az adatbázisban tárolt adatok nagy részét.

A 3.3. ábra két egyedet és az attribútumaik értékeit ábrázolja. Az e_1 DOLGOZÓ egyednek négy attribútuma van: Név, Cím, Életkor és Otthoni telefonszám; az értékeik rendre 'Kovács János', '4032 Debrecen, Virág utca 18.', '55' és '52/123456'. A c_1 VÁLLALAT egyednek három attribútuma van: Név, Székhely és Elnök; az értékeik rendre 'Fiastyúk Bt.', 'Pusztakotkodács' és 'Kovács János'.

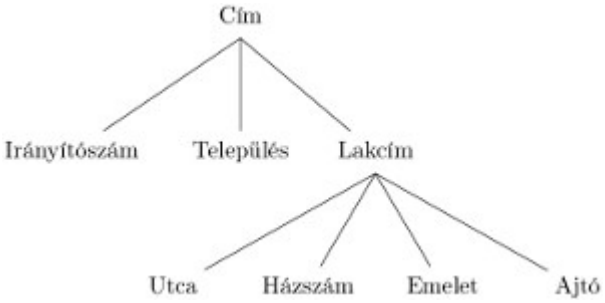
3.3. ábra - Két egyed, az e_1 DOLGOZÓ és a c_1 VÁLLALAT egyed, és az ő attribútumaik.



Az ER modellben különböző fajta attribútumok léteznek: *egyszerű* vagy *összetett*, *egyértékű* vagy *többértékű*, illetve *tárolt* vagy *származtatott* attribútumok. Először definiáljuk ezeket az attribútumtípusokat, és példák segítségével illusztráljuk őket. Utána bevezetjük a *NULL érték* fogalmát egy attribútumra vonatkozóan.

Összetett és egyszerű (atomi) attribútumok. Az **összetett attribútumok** kisebb részekre bonthatók, amelyek több, egymástól független jelentéssel bíró elemi attribútumot reprezentálnak. Például a 3.3. ábrán bemutatott alkalmazott Cím attribútuma Irányítószám[2], Település és Lakcím attribútumokra bontható fel, '4032', 'Debrecen' és 'Virág utca 18.' értékekkel. Azokat az attribútumokat, amelyeket nem bontunk részekre, **egyszerű** vagy **atomi attribútumoknak** nevezzük. Az összetett attribútumok hierarchiát alkothatnak; például a Lakcím tovább bontható négy egyszerű attribútumra: Utca, Házszám, Emelet és Ajtó attribútumokra, ahogyan az a 3.4. ábrán látható. Egy összetett attribútum értéke az őt alkotó egyszerű attribútumok értékeinek a konkatenációja.

3.4. ábra - Összetett attribútumok egy hierarchiája.



Az összetett attribútumok nagyon hasznosak olyan szituációk modellezésénél, amelyekben a felhasználó néha egységként hivatkozik az összetett attribútumra, máskor viszont külön-külön hivatkozik annak komponenseire. Ha az összetett attribútumra csak mint teljes egészre hivatkozunk, akkor nincs szükség arra, hogy komponensekre bontsuk. Ha például nem kell hivatkozunk egy cím önálló komponenseire (Irányítószám, Utca stb.), akkor a teljes cím egy egyszerű attribútum lehet.

Egyértékű és többértékű attribútumok. A legtöbb attribútum egy egyedben csak egy értékkel rendelkezik; az ilyen attribútumokat **egyértékűnek** nevezzük. Például az Életkor egy személy egyértékű attribútuma. Bizonyos esetekben egy attribútum értékek egy halmazával rendelkezhet ugyanazon egyedben — például egy autó Színek attribútuma vagy egy személy Diplomák attribútuma. Egyszínű autók esetén a Színek attribútum egyetlen értékkel rendelkezik, míg a két színnel fényezett autók esetében két értékkel. Hasonlóan, egy személynek lehet, hogy egyetlen diplomája sincs, egy másik személynek egy, egy harmadiknak kettő vagy több diplomája lehet; ezáltal a Diplomák attribútum különböző személyeknél különböző *számú értéket* vehet fel. Az ilyen attribútumokat nevezzük **többértékűnek**. Egy többértékű attribútumnak lehet alsó és felső korlátja, amelyek behatárolják a különböző egyedeknél a felvehető értékek számát. Például az autó Színek attribútuma legalább egy és legfeljebb három értéket vehet fel, ha feltételezzük, hogy egy autó legfeljebb három színre fényezhető.

Tárolt és származtatott attribútumok. Bizonyos esetekben kettő (vagy több) attribútum értékei kapcsolatban állnak egymással — például egy személy Életkor és Születési idő attribtútumai. Egy konrét személy egyed esetén az Életkor értéke meghatározható az aktuális (mai) dátum és az adott személyhez tartozó Születési idő értékéből. Az Életkor attribútumot emiatt **származtatott attribútumnak** nevezzük, és azt mondjuk, hogy a Születési idő attribútumból **származtatható**, amelyet pedig **tárolt attribútumnak** nevezünk. Egyes attribútumértékek *az egyeddel kapcsolatban álló más egyedekből* származtathatók; például egy OSZTÁLY egyed Dolgozók_száma attribútuma úgy származtatható, hogy megszámoljuk az adott osztályhoz tartozó (azaz ott dolgozó) alkalmazottakat.

NULL értékek. Előfordulhat, hogy egy adott egyed valamelyik attribútumának nincs használható értéke. Például egy cím Ajtó attribútuma csak azon címek esetén játszik szerepet, ahol a lakásokat ajtónként számozzák; másfajta épületeknél, mint például a családi házaknál, nem. Hasonlóan, a Diplomák attribútum csak diplomával rendelkező személyeknél érdekes. Az ilyen helyzetekben egy speciális értéket alkalmazunk, amelyet NULL értéknek nevezünk. A családi házak címében az Ajtó attribútum, egy diplomával nem rendelkező személy esetén pedig a Diplomák attribútum lesz NULL értékű. A NULL értéket használjuk akkor is, ha nem ismerjük egy adott egyed valamelyik attribútumának az értékét — például ha nem ismerjük a 3.3. ábrán látható 'Kovács János' otthoni telefonszámát. A NULL érték előbbi formájának a jelentése az, hogy *nem alkalmazható*, míg az utóbbié az, hogy *ismeretlen*. Az *ismeretlen* kategória további két esetre osztható. Az egyik esetben tudjuk, hogy az attribútumérték létezik, de *hiányzik* — például NULL értékű egy személy Magasság attribútuma. A másik esetben *nem tudjuk*, hogy az adott attribútumérték létezik-e — például NULL értékű egy személy Otthoni_telefonszám attribútuma.

Komplex attribútumok. Vegyük észre, hogy az összetett és a többértékű attribútumok tetszőlegesen egymásba ágyazhatók. Ezt a tetszőleges egymásba ágyazást úgy reprezentálhatjuk,

hogy az összetett attribútumok komponenseit kerek zárójelek között csoportosítjuk, és a komponenseket vesszővel választjuk el; a többértékű attribútumokat pedig kapcsos zárójelek közé írjuk. Az ilyen attribútumokat **komplex attribútumoknak** nevezzük. Például ha egy személynek egynél több lakása lehet, és minden lakásnak több telefonja, akkor a személy Cím_Telefon attribútuma a 3.5. ábrán látható módon adható meg.[3] A Telefon és Cím maguk is mindketten összetett attribútumok.

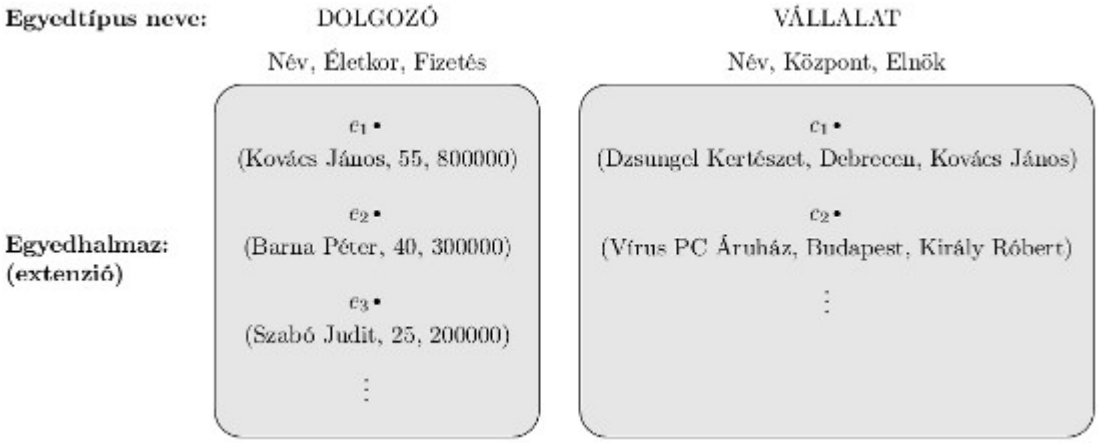
3.5. ábra - Egy komplex attribútum: a Cím_Telefon.

```
{ Cím_Telefon( Cím( Irányítószám, Település,
                    Lakcím( Utca, Házszám, Emelet, Ajtó ) ),
               { Telefon( Körzetszám, Telefonszám ) } ) }
```

Egyedtípusok, egyedhalmazok, kulcsok és értékkészletek (tartományok)

Egyedtípusok és egyedhalmazok. Egy adatbázis általában hasonló egyedek csoportjait tartalmazza. Egy dolgozók százait alkalmazó vállalat vélhetően hasonló információkat szeretne tárolni minden egyes dolgozójáról. Ezek a dolgozó egyedek ugyanazon attribútumokkal rendelkeznek, de minden egyed *saját érték(ek)et* vesz fel minden attribútumán. Egy **egyedtípus** olyan egyedek egy *kollekcióját* (vagy *halmazát*) definiálja, amelyek azonos attribútumokkal rendelkeznek. Az adatbázisban minden egyedtípus a nevével és az attribútumaival van megadva. A 3.6. ábra két egyedtípust ábrázol: a DOLGOZÓ-t és a VÁLLALAT-ot, valamint mindkettőhöz egy-egy attribútumlistát. Mindkét típus néhány konkrét egyede is látható rajta, az attribútumaik értékeivel együtt. Az adatbázis egy konkrét egyedtípusa összes egyedének egy adott időpillanatban vett kollekcióját egyedhalmaznak nevezzük; Az egyedhalmazra általában ugyanazzal a névvel hivatkozunk, mint az egyedhalmazra. Például a DOLGOZÓ egyaránt hivatkozik az *egyedtípusra* és az adatbázis *összes dolgozó egyedének* aktuális halmazára.

3.6. ábra - Két egyedtípus, a DOLGOZÓ és a VÁLLALAT, valamint néhány egyedük.



Az egyedtípusokat az ER diagramokon (lásd a 3.2. ábrát) egy téglalappal reprezentáljuk, amelybe beleírjuk az egyedtípus nevét. Az attribútumneveket oválisokba írjuk, és egyenes vonallal kötjük az egyedtípusukhoz. Az összetett attribútumok a komponenseikhez szintén egyenes vonallal kapcsolódnak. A többértékű attribútumokat dupla szegélyű oválisokba írjuk. A 3.7. (a) ábra az AUTÓ egyedtípust ábrázolja ezzel a jelöléssel.

3.7. ábra - Az AUTÓ egyedtípus két kulcs attribútummal, a Rendszámmal és az Alvázszámmal. (a) ER diagram jelölés. (b) Egyedhalmaz három egyeddel.

(a)

(b)

AUTÓ
Rendszám (Országkód, Jelsorozat), Alvázszám, Márka, Modell, Év, {Szín}

$AUTÓ_1$
((HU, HKL-008), TK629, Bugatti Veyron, nyitható tetejű, 2004, {vörös, fekete})

$AUTÓ_2$
((SK, BL-976AA), WP9872, Volkswagen Golf, 4 ajtós, 2005, {kék})

$AUTÓ_3$
((HU, KKD-006), TD729, Suzuki Swift, 4 ajtós, 2002, {fehér, kék})

⋮

Egy egyedtípus az azonos szerkezetű *egyedek halmazának* a **sémáját** vagy **intenzióját** írja le. Egy konkrét egyedtípus egyedeinek kollekcióját egy egyedhalmazba csoportosítjuk, amelyet az egyedtípus **extenziójának** is nevezünk.

Az egyedtípusok kulcs attribútumai. Az egyedtípusok egyedeinek egy fontos megszorítása az attribútumokon értelmezett **kulcs** vagy **egyediségi megszorítás**. Az egyedtípusok általában rendelkeznek egy olyan attribútummal, amelynek az értékei különbözőek az egyedhalmaz minden egyes egyede esetén. Az ilyen attribútumot **kulcs attribútumnak** nevezzük, értékeit pedig az egyes egyedek egyedi azonosítására használhatjuk. A 3.6. ábrán szereplő VÁLLALAT egyedtípusnak például a Név attribútum kulcsa, mivel két vállalatnak nem lehet ugyanaz a neve. A SZEMÉLY egyedtípusnak az Szsz (személyi szám) a tipikus kulcs attribútuma. Néha több attribútum együtt alkot kulcsot, ami azt jelenti, hogy az attribútumértékek *kombinációjának* kell különbözőnek lenni minden egyes egyed esetén. Ha egy attribútumhalmaz ilyen tulajdonsággal bír, akkor ezt az ER modellben úgy kell helyesen reprezentálni, hogy definiálunk egy *összetett attribútumot*, és azt jelöljük meg az egyedtípus kulcs attribútumaként. Ügyeljünk rá, hogy az ilyen összetett kulcsnak *minimálisnak* kell lennie; azaz az összetett attribútumnak minden komponens attribútumot tartalmaznia kell ahhoz, hogy az egyediségi tulajdonsága fennálljon. A kulcs nem tartalmazhat fölösleges attribútumokat. Az ER diagramokon a kulcs attribútumokat úgy jelöljük, hogy a nevüket **aláhúzzuk** az oválisban, ahogy a 3.7. (a) ábrán látható.

Az, hogy egy attribútumot egy egyedtípus kulcsaként definiálunk, azt jelenti, hogy a fent említett egyediségi tulajdonságnak az egyedtípus *minden egyes egyedhalmazára* fenn kell állnia. Emiatt ez egy olyan megszorítás, amely tiltja, hogy két egyed egyidejűleg ugyanazon értékkel rendelkezzen a kulcs attribútumon. Ez nem egy konkrét extenzió tulajdonsága, hanem az egyedtípus *összes extenziójának* egy megszorítása. Ez a kulcs megszorítás (és más megszorítások is, amelyeket később tárgyalunk) az adatbázis által reprezentált minivilág megszorításaiból származik.

Egyes egyedtípusok *egynél több* kulcs attribútummal rendelkeznek. Az AUTÓ egyedtípusnak (3.7. ábra) például a Rendszám és az Alvázszám attribútuma is önmagában kulcsa. A Rendszám attribútum jó példa az összetett kulcsra, amely két egyszerű komponens attribútumból, az Államból és a Jelsorozatból tevődik össze, amelyek közül önmagában egyik sem kulcs. Az is előfordulhat, hogy egy egyedtípusnak *nincs kulcsa*, ebben az esetben *gyenge egyedtípusnak* nevezzük (lásd a .

alfejezetet).

Az attribútumok értékkészlete (tartománya). Az egyedtípusok minden egyszerű attribútumához kapcsolódik egy **értékkészlet** (vagy **értéktartomány**), amely azon értékek halmaza, amelyeket az attribútumhoz hozzárendelhetünk az egyes konkrét egyedekben. Ha a 3.6. ábrán a dolgozók életkora 16 és 70 közé eshet, akkor a DOLGOZÓ Életkor attribútumának értékkészletét a 16 és 70 közötti egész számok halmazaként definiálhatjuk. Hasonlóan a Név attribútum értékkészletét szóköz karakterekkel elválasztott, alfabetikus karakterekből álló sztringként definiálhatjuk, stb. Az értékkészleteket nem tüntetjük fel az ER diagramokon. Az értékkészleteket tipikusan az alapvető **adattípusokkal** adjuk meg, amelyek a legtöbb programozási nyelvben elérhetők, mint például az egész, a sztring, a logikai, a valós, a felsorolásos típus, az intervallum típus stb. Dátumot, időt és más fogalmakat reprezentáló további adattípusokat is alkalmazhatunk.

Matematikailag egy E egyedtípus A attribútumát, amelynek az értékkészlete V, egy E-ből V-nek a $P(V)$ hatványhalmazába[4] képező **függvényként** definiálhatjuk:

- $A : E \rightarrow P(V)$

Az e egyed A attribútumának az értékére $A(e)$ formában hivatkozunk. Az előző definíció az egyértékű és többértékű attribútumokat, valamint a NULL értékeket is lefedi. A NULL értéket az *üres halmaz* reprezentálja. Egyértékű attribútumok esetén az $A(e)$ egyelemű halmaz lesz az E minden e egyede esetén, míg többértékű attribútumok esetén nincs semmilyen megkötés.[5] Egy A összetett attribútum esetén a V értékkészlet a $P(V_1)$, $P(V_2)$, …, $P(V_n)$ Descartes-szorzata, ahol V_1, V_2, …, V_n az A-t alkotó egyszerű komponens attribútumok értékkészletei:

- $V = P(V_1) \times P(V_2) \times \ldots \times P(V_n)$

Az értékkészlet minden lehetséges értéket tartalmaz. Rendszerint ezen értékek közül csak kevés található meg az adatbázisban, azok, amelyek a minivilág állapotát leíró adatokat reprezentálják. Ezek felelnek meg azoknak az adatoknak, amelyek a minivilágban ténylegesen léteznek.

A VÁLLALAT adatbázis kezdeti koncepcionális terve

Ezek után definiálhatjuk a VÁLLALAT adatbázis egyedtípusait a . alfejezetben leírt követelmények alapján. Most definiálunk néhány egyedtípust és az attribútumaikat, majd a . alfejezetben finomítjuk a tervünket, miután bevezetjük a kapcsolat fogalmát. A . alfejezetben felsorolt követelmények alapján négy egyedtípust azonosíthatunk, egyet-egyet a specifikációban megadott négy pont mindegyikéhez (lásd a 3.8. ábrát):

1. Az OSZTÁLY egyedtípust Név, Szám, Helyszínek, Osztályvezető és Kezdő_dátum attribútumokkal. A Helyszínek az egyetlen többértékű attribútum. Megadhatjuk, hogy mind a Név, mind a Szám (külön) kulcs attribútumok legyenek, mivel mindkettőt egyedinek definiáltuk.
2. A PROJEKT egyedtípust Név, Szám, Helyszín és Irányító_osztály attribútumokkal. Mind a Név, mind a Szám (külön) kulcs attribútumok.
3. A DOLGOZÓ egyedtípust Név, Szsz, Nem, Lakcím, Fizetés, Szdátum, Osztály és Főnöke attribútumokkal. Mind a Név, mind a Lakcím lehet összetett attribútum; ez azonban nem volt megadva a követelmények között. Vissza kell mennünk a felhasználókig, hogy lássuk, hogy bármelyikük is fog-e hivatkozni a Név egyes komponenseire (vezetéknév, keresztnév) vagy a Lakcím egyes komponenseire.
4. A HOZZÁTARTOZÓ egyedtípust Dolgozó, Hozzátartozó_neve, Nem, Szdátum és (a dolgozóval vett) Kapcsolat attribútumokkal.

3.8. ábra - Egyedtípusok előzetes terve a VÁLLALAT adatbázishoz. A feltüntetett

attribútumok némelyikét a későbbiekben kapcsolatokká fogjuk finomítani.

Eddig nem reprezentáltuk azt a tényt, hogy egy dolgozó több projekten is dolgozhat, mint ahogy azt sem, hogy egy dolgozó hány órát dolgozik hetente az egyes projekteken. Ezek a jellemzők a . alfejezetben felsorolt harmadik követelmény részeként voltak megadva, és a DOLGOZÓ Dolgozik_rajta nevű többértékű összetett attribútumával reprezentálhatók, amely a (Projekt, Órák) egyszerű komponensekkel rendelkezik. Egy másik lehetőség lenne, hogy a PROJEKT Dolgozók nevű többértékű összetett attribútumával reprezentáljuk őket a (Dolgozó, Órák) egyszerű komponensekkel. A 3.8. ábrán, amely a fent leírt egyedtípusokat ábrázolja, az első megoldást választottuk. A DOLGOZÓ Név attribútuma az ábrán összetett attribútumként látható, feltehetően a felhasználókkal történt konzultáció hatására.

Kapcsolattípusok, kapcsolathalmazok, szerepkörök és strukturális megszorítások

A 3.8. ábrán számos *implicit kapcsolatot* láthatunk különböző egyedtípusok között. Valójában valahányszor egy egyedtípus egy attribútuma egy másik egyedtípusra hivatkozik, valamilyen kapcsolat áll fenn. Az OSZTÁLY Osztályvezető attribútuma például egy olyan dolgozóra hivatkozik, aki vezeti az osztályt; a PROJEKT Irányító_osztály attribútuma arra az osztályra hivatkozik, amelyik irányítja a projektet; a DOLGOZÓ Főnöke attribútuma egy másik dolgozóra

hivatkozik (arra, aki ennek a dolgozónak a főnöke); a DOLGOZÓ Osztály attribútuma arra az osztályra hivatkozik, ahol a dolgozó dolgozik; és így tovább. Az ER modellben ezeket a hivatkozásokat nem attribútumokként, hanem **kapcsolatokként** érdemes reprezentálni, amelyekről ebben a fejezetben lesz szó. A VÁLLALAT adatbázissémát finomítani fogjuk a . alfejezetben úgy, hogy a kapcsolatokat explicit módon ábrázoljuk. Az egyedtípusok kezdeti tervében a kapcsolatokat tipikusan attribútumok formájában jelenítjük meg. Ahogy a tervet finomítjuk, ezeket az attribútumokat átalakítjuk az egyedtípusok közötti kapcsolatokká.

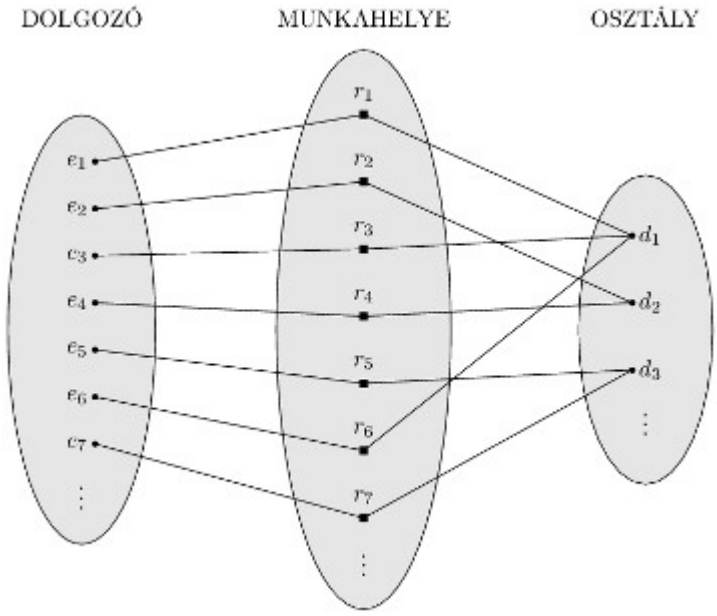
Ez az alfejezet a következőképpen épül föl: a . alfejezet bevezeti a kapcsolattípus, kapcsolathalmaz és kapcsolat-előfordulás fogalmait. A . alfejezetben definiáljuk a kapcsolat foka, a szerepkörnév és a rekurzív kapcsolat fogalmát, majd a . alfejezetben a kapcsolatok strukturális megszorításairól — például a számosságról és a létezésfüggőségről — tárgyalunk. A . alfejezet azt mutatja be, hogy hogyan lehetnek a kapcsolattípusoknak is attribútumaik.

Kapcsolattípusok, -halmazok és -előfordulások

Az E_1, E_2, …hellip;, E_n-nel jelölt n darab egyedtípus közötti R **kapcsolattípus** egy megfeleltetés-halmazt — más szóval egy **kapcsolathalmazt** — definiál az ezen egyedtípusokból származó egyedek között. Mint az egyedtípusok és az egyedhalmazok esetén, egy kapcsolattípusra és a neki megfelelő kapcsolathalmazra *ugyanazzal a névvel*, R-rel szokás hivatkozni. Matematikailag az R kapcsolathalmaz r_i **kapcsolatelőfordulások** egy halmaza, ahol minden r_i n darab egyedet (e_1, e_2, …, e_n-et) kapcsol össze, és minden r_i-beli e_j egyed az E_j egyedtípus eleme, $1 \leq j \leq n$. Ennélfogva egy kapcsolattípus értelmezhető E_1, E_2, …, E_n fölötti matematikai relációként; azaz definiálható az $E_1 \times E_2 \times \ldots \times E_n$ Descartes-szorzat egy részhalmazaként. Az E_1, E_2, …, E_n egyedtípusokról azt mondjuk, hogy **részt vesznek** az R kapcsolattípusban; és hasonlóan, az e_1, e_2, …, e_n egyedekről azt mondjuk, hogy részt vesznek az $r_i = (e_1, e_2, \ldots, e_n)$ kapcsolatelőfordulásban.

Informálisan az R-beli r_i kapcsolat-előfordulások egyedek olyan asszociációi, amelyek minden részt vevő egyedtípusból pontosan egy egyedet tartalmaznak. Minden ilyen r_i kapcsolatelőfordulás azt a tényt fejezi ki, hogy az r_i-ben részt vevő egyedek valamilyen módon kapcsolatban állnak egymással a megfelelő minivilágbeli szituációban. Tekintsük például a DOLGOZÓ és az OSZTÁLY egyedtípusok között értelmezett MUNKAHELYE kapcsolattípust, amely minden dolgozóhoz hozzárendeli azt az osztályt, ahol az illető dolgozik. A MUNKAHELYE kapcsolathalmazbeli minden egyes kapcsolat-előfordulás egy DOLGOZÓ egyedet és egy OSZTÁLY egyedet rendel egymáshoz. A 3.9. ábra illusztrálja ezt a példát, amelyen az r_i kapcsolat-előfordulások láthatók, összekötve az r_i-ben részt vevő DOLGOZÓ és OSZTÁLY egyedekkel. A 3.9. ábra által reprezentált minivilágban az e_1, e_3 és e_6 dolgozók a d_1 osztályon dolgoznak; az e_2 és e_4 dolgozók a d_2 osztályon dolgoznak; az e_5 és e_7 dolgozók pedig a d_3 osztályon dolgoznak.

3.9. ábra - Néhány előfordulás a MUNKAHELYE kapcsolathalmazból, amelyek egy MUNKAHELYE kapcsolattípust reprezentál a DOLGOZÓ és az OSZTÁLY között.

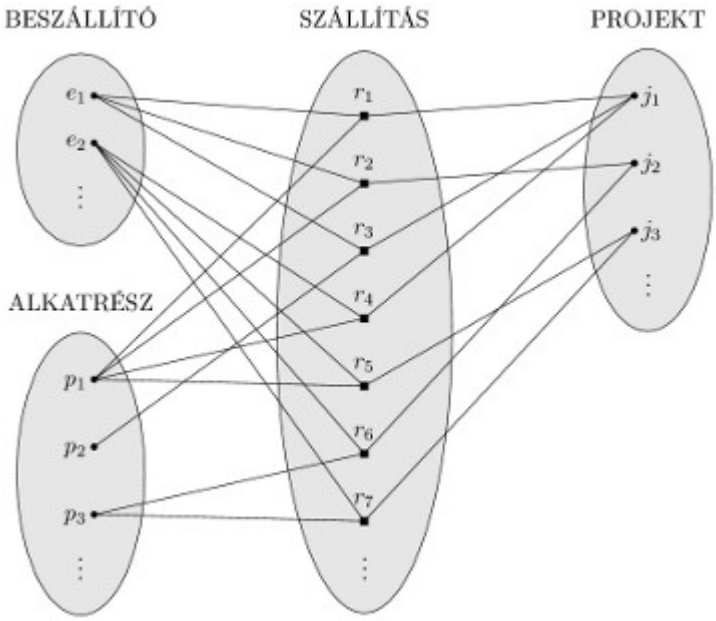


Az ER sémákban a kapcsolattípusokat rombusz alakú dobozokkal ábrázolják, amelyek folytonos egyenes vonalakkal kapcsolódnak a kapcsolatban résztvevő egyedtípusokat reprezentáló téglalap alakú dobozokhoz. A kapcsolat nevét a rombusz alakú dobozba írják (lásd 3.2. ábra).

A kapcsolat foka, szerepkörnevek és rekurzív kapcsolatok

Kapcsolattípus foka. Egy kapcsolattípus **foka** a kapcsolatban részt vevő egyedtípusok száma. Így például a MUNKAHELYE kapcsolat foka kettő. A másodfokú kapcsolatot **bináris**, a harmadfokút **ternáris** kapcsolatnak hívjuk. A 3.10. ábrán bemutatott SZÁLLÍTÁS kapcsolat például egy ternáris kapcsolat, ahol minden egyes r_i kapcsolatelőfordulás három egyedet — egy *s* szállítót, egy *p* alkatrészt és egy *j* projektet — kapcsol össze, valahányszor egy *s* szállító egy *p* alkatrészt szállít egy *j* projekthez. A kapcsolatok általában bármilyen fokúak lehetnek, de a leggyakoribbak a bináris kapcsolatok. A magasabb fokú kapcsolatok általában komplexebbek, mint a bináris kapcsolatok; a . alfejezetben fogjuk őket tovább tárgyalni.

3.10. ábra - Néhány kapcsolat-előfordulás a SZÁLLÍTÁS ternáris kapcsolathalmazból.

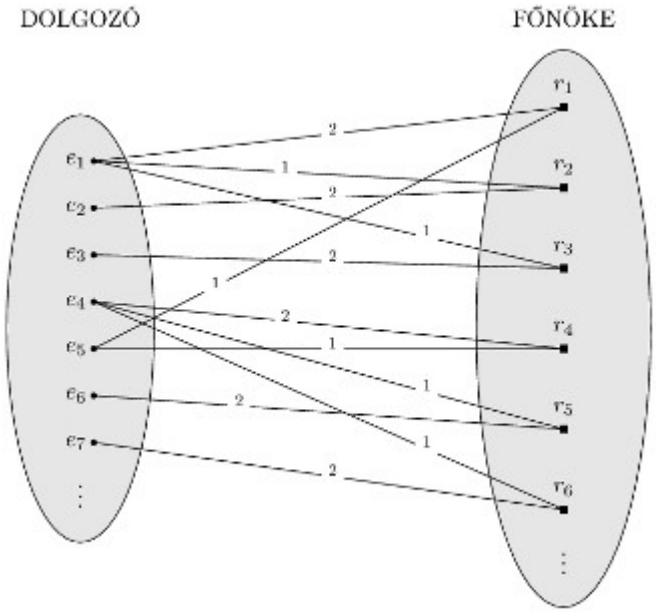


Kapcsolatok mint attribútumok. Néha az a kényelmes, ha a kapcsolattípusokra mint attribútumokra gondolunk, ahogy azt a . alfejezetben is tettük. Tekintsük a 3.9. ábrán látható MUNKAHELYE kapcsolattípust. Elképzelhetünk egy attribútumot a DOLGOZÓ egyedtípusban, amelyet Osztálynak hívunk, és amelynek értéke minden egyes alkalmazott egyed esetén az az OSZTÁLY egyed (vagy egy hivatkozás rá), ahol az alkalmazott dolgozik. Így ennek az Osztály attribútumnak az értékhalmaza az *összes* OSZTÁLY egyed halmaza, amely azonos az OSZTÁLY egyedhalmazzal. Ezt tettük a 3.8. ábrán, amikor a VÁLLALAT adatbázis DOLGOZÓ egyedtípusának kiinduló tervét adtuk meg. Ha azonban egy bináris kapcsolatra attribútumként gondolunk, mindig két lehetőségünk van. Ebben a példában a másik lehetőség az, hogy az OSZTÁLY egyedtípusban képzelünk el egy Alkalmazottak többértékű attribútumot, amelynek az értéke minden egyes OSZTÁLY egyed esetén azon DOLGOZÓ egyedek halmaza, akik az adott osztályon dolgoznak. Ennek az Alkalmazottak attribútumnak az értékhalmaza a DOLGOZÓ egyedhalmazának hatványhalmaza. Az említett két attribútum — a DOLGOZÓ Osztály attribútuma vagy az OSZTÁLY Alkalmazottak attribútuma — egyaránt alkalmas a MUNKAHELYE kapcsolattípus reprezentálására. Ha mindkettőt reprezentáljuk, akkor egymás inverzeinek kell lenniük.[6]

Szerepkörnevek és rekurzív kapcsolatok. Minden egyedtípus, amely részt vesz egy kapcsolattípusban, egy konkrét **szerepet** játszik a kapcsolatban. A **szerepkörnév** jelöli azt a szerepet, amelyet az egyedtípusnak a kapcsolatban részt vevő egyedei játszanak az egyes kapcsolatelőfordulásokban, és segít megérteni, mit is jelent a kapcsolat. Például a MUNKAHELYE kapcsolattípusban a DOLGOZÓ az *alkalmazott* vagy *dolgozó* szerepét, az OSZTÁLY pedig az *osztály* vagy *munkáltató* szerepét játssza.

A szerepkörnevek technikailag nem szükségesek azokban a kapcsolattípusokban, ahol az összes részt vevő egyedtípus különböző, mivel minden részt vevő egyedtípus neve használható szerepkörnévként. Bizonyos esetekben azonban *ugyanazon* egyedtípus egynél többször vesz részt egy kapcsolattípusban *különböző szerepekben*. Ezekben az esetekben szükségessé válik a szerepkörnév, hogy megkülönböztessük az egyes részvételek jelentéseit. Az ilyen kapcsolattípusokat **rekurzív kapcsolatoknak** nevezzük. A 3.11. ábra mutat erre egy példát. A FŐNÖKE kapcsolattípus összeköt egy dolgozót egy főnökkel, ahol mind az alkalmazott, mind a főnök egyedek ugyanazon DOLGOZÓ egyedtípusból valók. Így a DOLGOZÓ egyedtípus *kétszer vesz részt* a FŐNÖKE kapcsolatban: egyrészt a *főnök*, másrészt a *beosztott* (vagy *alárendelt*) szerepében. A FŐNÖKE minden r_i kapcsolatelőfordulása két dolgozó egyedet kapcsol össze (e_j-t és e_k-t), amelyek közül az egyik játssza a főnök szerepét, a másik pedig a beosztottét. A 3.11. ábrán az 1-essel jelölt vonalak jelölik a főnök szerepkört, a 2-essel jelöltek pedig a beosztott szerepkört; így e_1 főnöke e_2-nek és e_3-nak, e_4 főnöke e_6-nak és e_7-nek, e_5 főnöke e_1-nek és e_4-nek. Ebben a példában minden kapcsolat-előfordulásnak két vonallal kell rendelkeznie, egy 1-essel jelöltnek (főnök) és egy 2-essel jelöltnek (beosztott).

3.11. ábra - A FŐNÖKE rekurzív kapcsolat a *főnök* szerepkörű DOLGOZÓ (1) és a *beosztott* szerepkörű DOLGOZÓ (2) között.

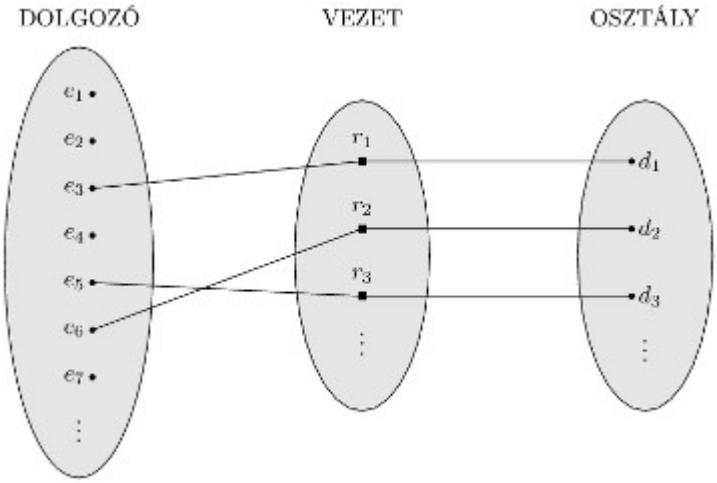


A kapcsolattípusok megszorításai

A kapcsolattípusok általában rendelkeznek bizonyos megszorításokkal, amelyek korlátozzák azokat a lehetséges egyedkombinációkat, amelyek részt vehetnek a megfelelő kapcsolathalmazban. Ezeket a megszorításokat a kapcsolatok által reprezentált minivilágbeli szituáció alapján határozzuk meg. Ha például a 3.9. ábrán a vállalatnak van egy olyan szabálya, hogy minden dolgozónak pontosan egy osztályon kell dolgoznia, akkor ezt a megszorítást szeretnénk a sémán is megjeleníteni. A kapcsolatokra vonatkozó megszorításoknak két fő típusát különböztetjük meg: *számosság* és *részvétel*.

A bináris kapcsolatok számossága. A bináros kapcsolatok **számossága** meghatározza azon kapcsolat-előfordulások *maximális* számát, amelyekben egy egyed részt vehet. A MUNKAHELYE bináris kapcsolattípus esetén például az OSZTÁLY:DOLGOZÓ 1:N számosságú, ami azt jelenti, hogy minden osztály akárhány dolgozóhoz kapcsolódhat (azaz tetszőleges számú dolgozót alkalmazhat),[7] viszont egy dolgozó csak egy osztályhoz kapcsolódhat (azaz csak egy osztályon dolgozhat). A bináris kapcsolattípusok lehetséges számosságai: 1:1, 1:N, N:1 és M:N.

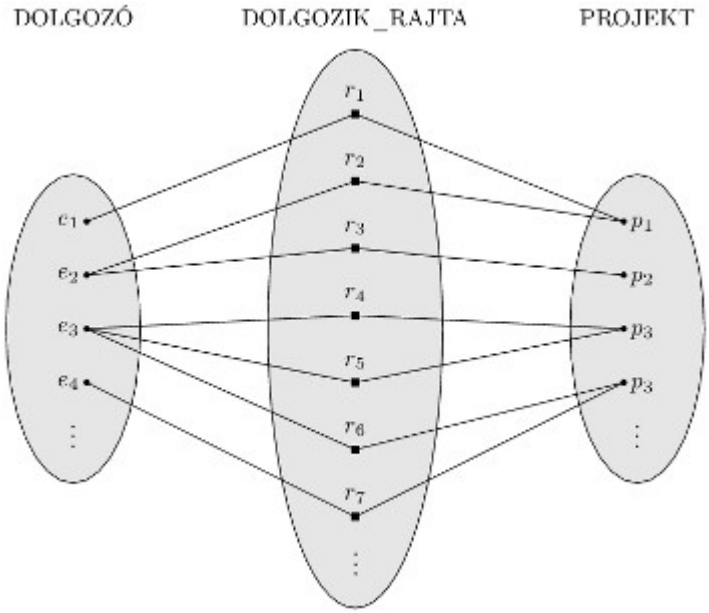
3.12. ábra - Az 1:1 számosságú VEZETI kapcsolat.



1:1 számosságú bináris kapcsolatra példa a VEZETI (3.12. ábra), amely az egyes osztály egyedeket összekapcsolja azzal a dolgozóval, aki az adott osztályt vezeti. Ez azokat a minivilágbeli

megszorításokat reprezentálja, hogy — bármely időpillanatban — egy dolgozó csak egy osztályt vezethet, és egy osztálynak csak egy vezetője lehet. A DOLGOZIK_RAJTA kapcsolattípus (3.13. ábra) M:N számosságú, mert a minivilág szabálya szerint egy dolgozó akárhány projekten dolgozhat, és egy projekthez akárhány dolgozó tartozhat.

3.13. ábra - Az M:N számosságú DOLGOZIK_RAJTA kapcsolat.



A bináris kapcsolatok számosságát az ER diagramokon úgy reprezentáljuk, hogy 1-est, M-et vagy N-et írunk a rombuszok mellé, ahogy a 3.2. ábrán látható.

Részvételi megszorítások és létezésfüggőség. A **részvételi megszorítás** megadja, hogy egy egyed létezése függ-e attól, hogy kapcsolatban áll-e egy másik, a kapcsolattípuson keresztül elérhető egyeddel. Ez a megszorítás meghatározza azon kapcsolat-előfordulások *minimális* számát, amelyekben az egyes egyedek részt vehetnek, ezért néha **minimum számossági megszorításnak** is nevezzük. A részvételi megszorításoknak két típusa létezik (totális és részleges), amelyeket példákkal illusztrálunk. Ha egy vállalat előírásai kimondják, hogy *minden* dolgozót hozzá kell rendelni egy osztályhoz, akkor egy dolgozó egyed csak akkor létezhet, ha legalább egy MUNKAHELYE kapcsolat-előfordulásban részt vesz (3.9. ábra). A MUNKAHELYE kapcsolatban a DOLGOZÓ részvételét tehát **totális részvételnek** nevezzük, ami azt jelenti, hogy a dolgozó egyedek *teljes halmazában* minden egyednek kapcsolódnia kell egy osztály egyedhez a MUNKAHELYE kapcsolaton keresztül. A totális részvételt **létezésfüggőségnek** is nevezzük. A 3.12. ábrán nem várjuk el, hogy minden dolgozó vezessen egy osztályt, így a DOLGOZÓ részvétele a VEZETI kapcsolattípusban **részleges**, ami azt jelenti, hogy a dolgozó egyedek *halmazának csak egy része* kapcsolódik valamely osztály egyedhez a VEZETI kapcsolaton keresztül, de nem feltétlenül az összes. A számosságra és a részvételi megszorításokra együttesen a kapcsolattípusok **strukturális megszorításaiként** fogunk hivatkozni.

Az ER diagramokon a totális részvételt (vagy létezésfüggőséget) egy olyan *dupla vonallal* jelöljük, amely a részt vevő egyedtípust köti össze a kapcsolattal, míg a részleges részvételt *szimpla vonallal* reprezentáljuk (lásd a 3.2. ábrát).

A kapcsolattípusok attribútumai

A kapcsolattípusoknak is lehetnek attribútumaik, amelyek hasonlóak az egyedtípusokéihoz. Ha például szeretnénk feljegyezni, hogy egy dolgozó hetente hány órát dolgozik egy konkrét projekten, felvehetünk egy Órák attribútumot a 3.13. ábra DOLGOZIK_RAJTA kapcsolattípusához. Egy másik példa lehet, hogy felvesszük azt a dátumot, amikor egy osztályvezető megkezdte egy osztály

vezetését, a 3.12. ábra VEZETI kapcsolattípusának egy Kezdő_dátum elnevezésű attribútumaként.

Vegyük észre, hogy az 1:1 és az 1:N számosságú kapcsolattípusok attribútumait hozzácsatolhatjuk a részt vevő egyedtípusok egyikéhez. A VEZETI kapcsolat Kezdő_dátum attribútuma akár a DOLGOZÓ, akár az OSZTÁLY attribútuma is lehet, bár koncepcionálisan a VEZETI kapcsolathoz tartozik. Ez azért van, mert a VEZETI egy 1:1 számosságú kapcsolat, ezért minden osztály és minden dolgozó egyed *legfeljebb egy* kapcsolat-előfordulásban vesz részt. Emiatt a Kezdő_dátum attribútum értéke külön is meghatározható vagy a részt vevő osztály egyeden, vagy a részt vevő dolgozó (osztályvezető) egyeden keresztül.

Egy 1:N számosságú kapcsolattípus esetén a kapcsolat attribútuma *csak* a kapcsolat N oldali egyedtípusához csatolható. Ha például a 3.9. ábrán a MUNKAHELYE kapcsolat szintén rendelkezne egy Kezdő_dátum attribútummal, amely azt jelezné, hogy egy dolgozó mikor kezdett az adott osztályon dolgozni, akkor ezt az attribútumot a DOLGOZÓ egy attribútumaként is felvehetnénk. Ez azért van, mert minden dolgozó csak egy osztályon dolgozik, így a MUNKAHELYE kapcsolat legfeljebb egy előfordulásában vesz részt. Mind az 1:1, mind az 1:N kapcsolattípusok esetén az a döntés, hogy egy kapcsolat egy attribútumát hová helyezzük (azaz hogy a kapcsolattípushoz vagy az egyik részt vevő egyedtípushoz csatoljuk-e), a sématervező szubjektív döntése.

Az M:N kapcsolattípusok esetén bizonyos attribútumok csak a kapcsolat-előfordulásokban *részt vevő egyedek kombinációja* segítségével határozhatók meg, egyetlen egyed segítségével azonban nem. Az ilyen attribútumokat *a kapcsolat attribútumaiként kell definiálni*. Példa erre az M:N számosságú DOLGOZIK_RAJTA kapcsolat Órák attribútuma (3.13. ábra); az egy dolgozó által egy projektre fordított heti munkaórák számát egy dolgozó-projekt kombináció alapján tudjuk csak meghatározni, külön ezen egyedek egyikével azonban nem.

Gyenge egyedtípusok

Azokat az egyedtípusokat, amelyek nem rendelkeznek saját kulcs attribútumokkal, **gyenge egyedtípusoknak** nevezzük. Ezzel ellentétben azokat a **hagyományos egyedtípusokat**, amelyeknek van kulcs attribútumuk — és amelyek közé az összes eddig tárgyalt példa is tartozik — **erős egyedtípusoknak** nevezzük. A gyenge egyedtípushoz tartozó egyedek azonosítása egyrészt egy másik egyedtípus bizonyos egyedeinek, másrészt saját attribútumértékeik közül egynek a felhasználásával történik. Ezt a másik egyedtípust **azonosító** vagy **tulajdonos egyedtípusnak**[8] nevezzük, a gyenge egyedtípust a tulajdonosával összekötő kapcsolattípust pedig a gyenge egyedtípus[9] **azonosító kapcsolatának** hívjuk. A gyenge egyedtípus mindig *totális résztvevője* az azonosító kapcsolatának (létezésfüggőség), mert egy gyenge egyedet nem lehet azonosítani tulajdonos egyed nélkül. Nem minden létezésfüggőség eredményez azonban gyenge egyedtípust. Például egy JOGOSÍTVÁNY egyed nem létezhet egy kapcsolódó SZEMÉLY egyed nélkül, bár neki is van saját kulcsa (a Vezetői_engedély_száma), és ezért nem gyenge egyed.

Vegyük például a HOZZÁTARTOZÓ egyedtípust, amely a DOLGOZÓ egyedtípushoz kapcsolódik, és amelyet az egyes dolgozók hozzátartozóinak a tárolására használunk egy 1:N kapcsolattípuson keresztül (3.2. ábra). A HOZZÁTARTOZÓ attribútumai a Név (a hozzátartozó keresztneve), a Szdátum, a Nem és a Kapcsolat (a dolgozóval). *Két különböző dolgozó* két hozzátartozójának véletlenül lehet ugyanaz a Neve, Szdátuma, Neme és Kapcsolata, annak ellenére, hogy különböző egyedekről van szó. Ezért csak azután azonosíthatók különböző egyedekként, ha már meghatároztuk azokat a *konkrét alkalmazott egyedeket*, amelyekhez az egyes hozzátartozók kapcsolódnak. Azt mondjuk, hogy az egyes DOLGOZÓ egyedek tulajdonosai a hozzájuk kapcsolódó HOZZÁTARTOZÓ egyedeknek.

A gyenge egyedtípusoknak **részleges kulcsuk** van, amely azon attribútumok halmaza, amelyek egyértelműen azonosítják azokat a gyenge egyedeket, amelyek *ugyanazon tulajdonos egyed(ek)hez*

kapcsolódnak.[10] A példánkban, ha feltételezzük, hogy ugyanazon alkalmazott két hozzátartozójának nem lehet azonos a keresztneve, akkor a HOZZÁTARTOZÓ Név attribútuma a részleges kulcs. Legrosszabb esetben *a gyenge egyedtípus összes attribútumából* képzett összetett attribútum lesz a részleges kulcs.

Az ER diagramokban a gyenge egyedtípust dupla szegélyű téglalappal, az azonosító kapcsolatát dupla szegélyű rombusszal jelöljük (lásd a 3.2. ábrát). A részleges kulcs attribútumot szaggatott vagy pontozott vonallal húzzuk alá.

A gyenge egyedtípusokat néha komplex (összetett, többértékű) attribútumokként is reprezentálhatjuk. Az előző példában megadhatnánk a DOLGOZÓ egyedtípushoz egy többértékű Hozzátartozók attribútumot, amely összetett attribútum lenne Név, Születési idő, Nem és Kapcsolat attribútumokkal. Hogy melyik reprezentációt használjuk, azt az adatbázis tervezője dönti el. A gyenge egyedtípus reprezentációt célszerű használni akkor, ha sok attribútumunk van. Ha a gyenge egyedtípus az azonosító kapcsolattípusán kívül más kapcsolattípusokban is részt vesz, akkor *nem* célszerű komplex attribútumként modellezni.

Általánosságban gyenge egyedtípusok tetszőleges számú szintjét definiálhatjuk; a tulajdonos egyedtípus maga is lehet gyenge egyedtípus. Egy gyenge egyedtípus ráadásul egynél több azonosító egyedtípussal is rendelkezhet, így az azonosító kapcsolattípusa kettőnél magasabb fokú is lehet, ahogy azt a . alfejezetben majd látni fogjuk.

A VÁLLALAT adatbázis ER tervének finomítása

A 3.8. ábra adatbázistervét tovább finomíthatjuk, ha azokat az attribútumokat, amelyek kapcsolatokat reprezentálnak, kicseréljük kapcsolattípusokra. Az egyes kapcsolattípusok számossága és részvételi megszorítása a . alfejezetben felsorolt követelmények alapján kerül meghatározásra. Ha valamilyen számosság vagy függőség nem határozható meg a követelményekből, akkor a felhasználókat kell ismételten megkérdezni az ilyen strukturális megszorításoknak a meghatározása végett.

Példánkban a következő kapcsolattípusokat definiáljuk:

- VEZETI, egy 1:1 számosságú kapcsolattípus a DOLGOZÓ és az OSZTÁLY között. A DOLGOZÓ részvétele részleges. Az OSZTÁLY részvétele nem derül ki pontosan a követelményekből. Megkérdezzük tehát a felhasználót, aki azt mondja, hogy egy osztálynak mindig kell, hogy legyen egy vezetője, amelyből következik a teljes (totális) részvétel.[11] A Kezdő_dátum attribútumot ehhez a kapcsolattípushoz rendeljük hozzá.
- MUNKAHELYE, egy 1:N számosságú kapcsolattípus az OSZTÁLY és a DOLGOZÓ között. Mindkét részvétel teljes (totális).
- IRÁNYÍTJA, egy 1:N számosságú kapcsolattípus az OSZTÁLY és a PROJEKT között. A PROJEKT részvétele teljes (totális), míg az OSZTÁLY részvétele részleges lesz, miután a felhasználókkal történő konzultáció során kiderül, hogy lehetnek olyan osztályok, amelyek nem irányítanak projekteket.
- FŐNÖKE, egy 1:N számosságú kapcsolattípus a (főnök szerepkörű) DOLGOZÓ és a (beosztott szerepkörű) DOLGOZÓ között. Mindkét részvételt részlegesnek (parciálisnak) definiáljuk, miután a felhasználók jelzik, hogy nem minden dolgozó főnök és nem minden dolgozónak van főnöke.
- DOLGOZIK_RAJTA, egy M:N számosságúnak definiált kapcsolattípus az Órák attribútummal, miután a felhasználók jelzik, hogy egy projekten több dolgozó is dolgozhat. Mindkét részvételt teljesnek (totálisnak) definiáljuk.
- HOZZÁTARTOZÓJA, egy 1:N számosságú kapcsolattípus a DOLGOZÓ és a

HOZZÁTARTOZÓ között, amely egyúttal a HOZZÁTARTOZÓ gyenge egyedtípus azonosító kapcsolata is. A DOLGOZÓ részvétele részleges (parciális), míg a HOZZÁTARTOZÓ részvétele teljes (totális).

A fenti hat kapcsolattípus definiálása után eltávolítjuk a 3.8. ábra egyedtípusaiból az összes olyan attribútumot, amelyből kapcsolattípust készítettünk. Ilyen az Osztályvezető és a Kezdő_dátum az OSZTÁLY-ban; az Irányító_osztály a PROJEKT-ben; az Osztály, a Főnöke és a Dolgozik_rajta a DOLGOZÓ-ban; illetve a Dolgozó a HOZZÁTARTOZÓ-ban. Fontos, hogy a redundancia a lehető legkisebb legyen, amikor egy adatbázis koncepcionális sémáját tervezzük. Ha valamilyen redundancia mégis kívánatos a tárolási szinten vagy a felhasználói nézet szintjén, azt be lehet vezetni később, ahogyan erről a . alfejezetben már szóltunk.

ER diagramok, elnevezési konvenciók és tervezési kérdések

Az ER diagramokban használt jelölések

A 3.9-től 3.13-ig számozott ábrák az egyedtípusoknak a kapcsolattípusokban való részvételére mutatnak példákat az extenzióik megjelenítésével, azaz az egyedhalmazokban és kapcsolathalmazokban szereplő egyed-előfordulások és kapcsolat-előfordulások felsorolásával. Az ER diagramokon azonban a hangsúly nem a példányok, hanem a sémák reprezentálásán van. Ez sokkal hasznosabb az adatbázis-tervezés során, mivel az adatbázisséma ritkán változik, míg az egyedhalmazok tartalma gyakran. Ráadásul a sémát általában könnyebb megjeleníteni, mint az adatbázis extenzióját, mivel sokkal kisebb.

A 3.2. ábra a VÁLLALAT **ER adatbázissémáját** ábrázolja **ER diagramként**. Most áttekintjük az ER diagramok teljes jelölésrendszerét. Az egyedtípusokat, mint például a DOLGOZÓ, az OSZTÁLY és a PROJEKT, téglalapokkal ábrázoljuk. A kapcsolattípusokat, mint például a MUNKAHELYE, a VEZETI, az IRÁNYÍTJA és a DOLGOZIK_RAJTA, rombuszokkal jelöljük, amelyeket egyenes vonalakkal kötünk össze a részt vevő egyedtípusokkal. Az attribútumokat oválisokkal ábrázoljuk, és minden attribútumot egy egyenes vonal köt az egyedtípusához vagy a kapcsolattípusához. Az összetett attribútumok komponens attribútumait hozzákapcsoljuk az összetett attribútumot jelképező oválishoz, ahogy az a DOLGOZÓ Név attribútumánál látható. A többértékű attribútumokat dupla szegélyű oválisokba írjuk, ahogy az az OSZTÁLY Helyszínek attribútumánál látható. A kulcs attribútumok nevét aláhúzzuk. A származtatott attribútumokat szaggatott szegélyű oválisokba írjuk, ahogy az az OSZTÁLY Dolgozók_száma attribútumánál látható.

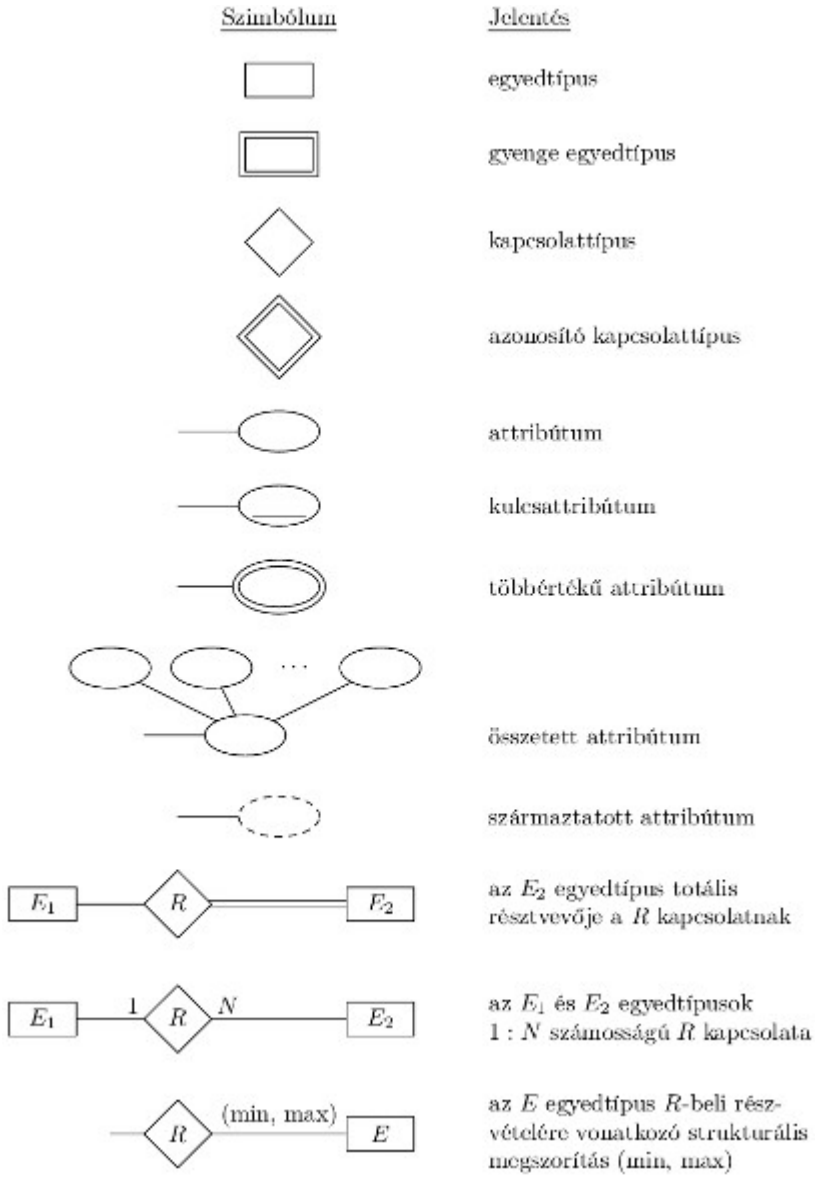
A gyenge egyedtípusokat úgy különböztetjük meg, hogy dupla szegélyű téglalapokba írjuk a nevüket, az azonosító kapcsolatukat pedig dupla szegélyű rombuszokba, ahogy az a HOZZÁTARTOZÓ egyedtípusnál és a HOZZÁTARTOZÓJA azonosító kapcsolattípusnál látható. A gyenge egyedtípus részleges kulcsát szaggatott vonallal húzzuk alá.

A 3.2. ábrán az összes *bináris* kapcsolattípus számosságát úgy jelöltük, hogy egy 1-est, egy M-et vagy egy N-et írtunk a kapcsolódó élekre. A VEZETI kapcsolatban az OSZTÁLY:DOLGOZÓ számossága 1:1, míg a MUNKAHELYE kapcsolatban az OSZTÁLY:DOLGOZÓ számossága 1:N, a DOLGOZIK_RAJTA kapcsolatban pedig a számosság M:N. A részvételi megszorítást úgy adjuk meg, hogy szimpla vonalat húzunk a részleges részvétel esetén, míg dupla vonalat a totális részvétel esetén (létezésfüggőség).

A 3.2. ábrán megadtuk a szerepkörneveket a FŐNÖKE kapcsolattípusnál, mert a DOLGOZÓ egyedtípus játssza mindkét szerepet a kapcsolatban. Vegyük észre, hogy a számosság 1:N a főnöktől a beosztott felé, mivel a beosztott szerepkörű dolgozóknak legfeljebb egy közvetlen főgöke van, míg a főnök szerepkörű dolgozók nulla vagy több dolgozónak lehetnek a főnökei.

A 3.14. ábra összefoglalja az ER diagramok jelölésrendszerét.

3.14. ábra - Az ER diagramok jelöléseinek összefoglalása.



A sémaelemek helyes elnevezése

Az adatbázisséma tervezése során az egyedtípusok, az attribútumok, a kapcsolattípusok és (különösen) a szerepkörök neveinek megválasztása nem mindig nyilvánvaló. Olyan neveket célszerű választani, amelyek a lehető legjobban utalnak a séma különböző konstrukcióihoz kapcsolódó jelentésre. Az egyedtípusok számára mi *egyes számú főneveket* választunk, nem pedig többes számút, mert az egyedtípus neve az adott egyedtípushoz tartozó minden egyes egyedre vonatkozik. Az ER diagramjainkban azt a konvenciót követjük, hogy az egyedtípusok és a kapcsolattípusok neveit csupa nagybetűvel, az attribútumnevek első betűit szintén nagybetűvel, a szerepkörneveket pedig csupa kisbetűvel írjuk. Ezt a konvenciót alkalmaztuk a 3.2. ábrán is.

Általános gyakorlat, hogy az adatbázis követelmények szövegszerű leírásában előforduló *főnevek* az egyedtípusok neveit, az *igék* pedig a kapcsolattípusok neveit jelzik. Az attribútumnevek általában azokból a további főnevekből származnak, amelyek az egyedtípusoknak megfelelő főneveket írják le.

Egy másik elnevezési szokás az, hogy a bináris kapcsolatok neveit úgy válasszuk meg, hogy a séma ER diagramja balról jobbra és felülről lefelé legyen olvasható. Általában mi is követtük ezt az

irányelvet a 3.2. ábrán.

Tervezési lehetőségek a koncepcionális tervezés során

Időnként nehéz eldönteni, hogy a minivilág egy konkrét fogalmát egyedtípusként, attribútumként vagy kapcsolattípusként célszerű-e modellezni. Ebben az alfejezetben adunk néhány rövid irányelvet arra vonatkozóan, hogy melyik konstrukciót célszerű választani konkrét szituációkban.

Általában a sématervezési folyamatot egy iteratív finomítási folyamatként kell tekinteni, ahol először egy kezdeti tervet hozunk létre, majd iteratívan finomítjuk azt, amíg meg nem kapjuk a legmegfelelőbb tervet. A gyakran használt finomítási lehetőségek közül az alábbiakban felsorolunk néhányat:

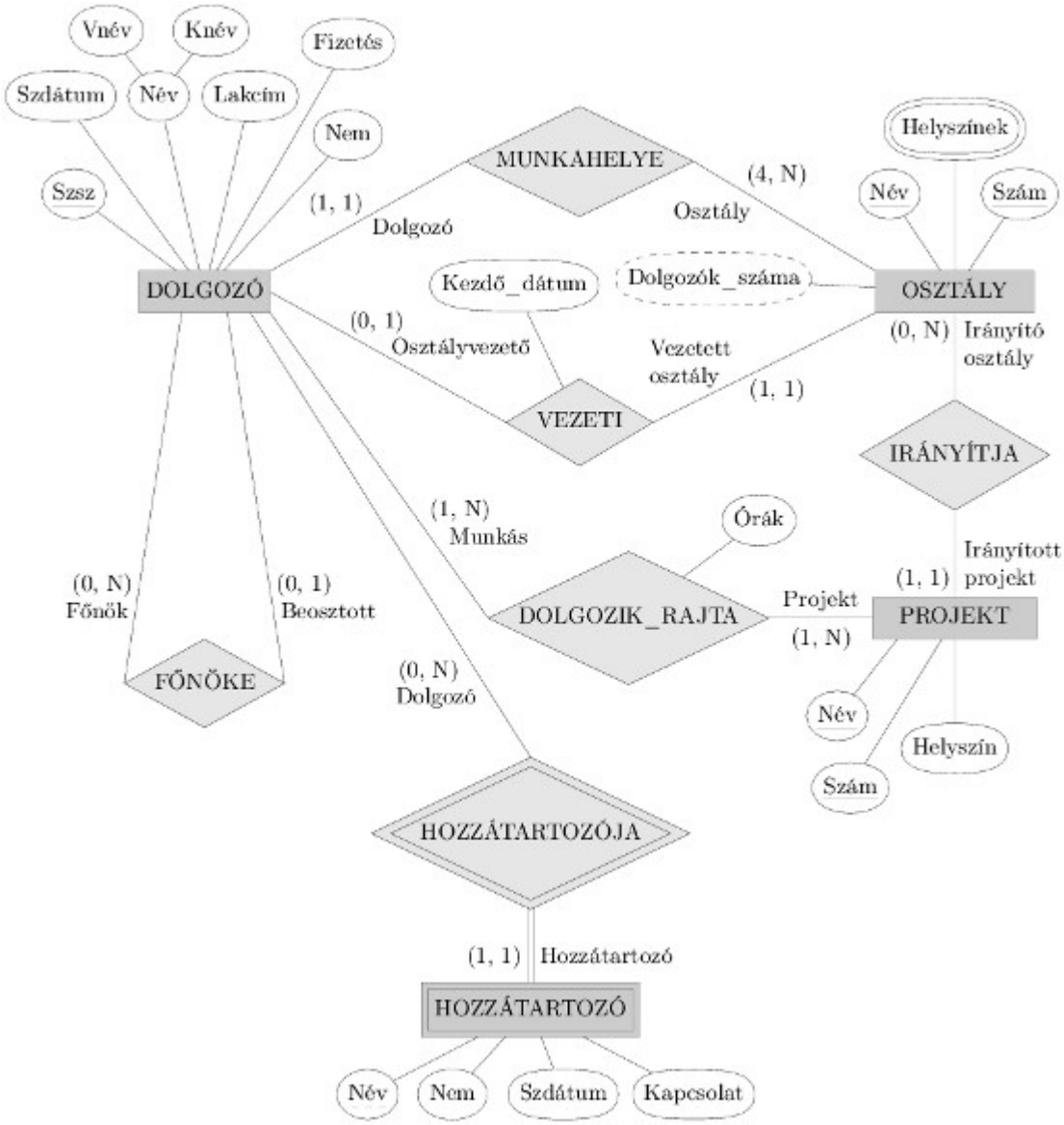
- Egy fogalmat először attribútumként modellezhetünk, amit aztán kapcsolattá finomíthatunk, ha azt tapasztaljuk, hogy az attribútum egy másik egyedtípusra hivatkozik. Gyakori eset, hogy az olyan attribútumpárok, amelyek egymás inverzei, bináris kapcsolattá finomíthatók. A finomításnak erről a fajtájáról részletesen tárgyaltunk a . alfejezetben.
- Hasonlóan, egy több egyedtípusban is létező attribútumot egy független egyedtípussá léptethetünk elő. Tegyük fel például, hogy az EGYETEM adatbázisban több egyedtípus (például HALLGATÓ, OKTATÓ és TANTÁRGY) is rendelkezik egy Tanszék attribútummal a kezdeti tervben; a tervező ekkor dönthet úgy, hogy létrehoz egy TANSZÉK egyedtípust egyetlen attribútummal (Tanszék_neve), amelyet megfelelő kapcsolatokon keresztül összekapcsol a három egyedtípussal (HALLGATÓ, OKTATÓ és TANTÁRGY). A TANSZÉK további attribútumait/kapcsolatait a későbbiek során is megadhatjuk.
- Az előző eset inverz finomítása is alkalmazható. Ha például az eredeti tervben létezik egy TANSZÉK egyedtípus egyetlen attribútummal (Tanszék_neve), és csak egyetlen másik egyedtípushoz, a HALLGATÓ-hoz kapcsolódik, akkor a TANSZÉK-et a HALLGATÓ egy attribútumává fokozhatjuk le.
- A . alfejezet tárgyalja a kapcsolatok fokára vonatkozó lehetőségeinket. A 4. fejezetben további finomításokról lesz szó a specializációval/generalizációval kapcsolatban.

Alternatív jelölések az ER diagramokon

Sok alternatív grafikus jelölés létezik az ER diagramok ábrázolására.

Ebben az alfejezetben egy, a kapcsolatok strukturális megszorításainak megadására használt alternatív ER jelölést mutatunk be. Ezt a jelölést használva egy (min, max) egészszám-párt írunk egy *R* kapcsolattípusban részt vevő *E* egyedtípus *részvételéhez*, ahol $0 \leq \min \leq \max$ és $\max \geq 1$. A számok azt jelentik, hogy minden *E*-beli *e* egyed esetén *e*-nek legalább min, legfeljebb max *R*-beli kapcsolatelőfordulásban kell részt vennie *bármely időpillanatban*. Ennél a módszernél min = 0 jelenti a részleges, míg min > 0 a totális részvételt.

3.15. ábra - A VÁLLALAT séma ER diagramja (min, max) jelölésű strukturális megszorításokkal és szerepkörnevekkel.



A 3.15. ábra a VÁLLALAT adatbázis sémáját ábrázolja a (min, max) jelölés használatával.[12] Rendszerint vagy a számosság/szimpla vonal/dupla vonal, *vagy* a (min, max) jelölést használjuk. A (min, max) jelölés pontosabb, és *tetszőleges fokú* kapcsolattípusok strukturális megszorításainak megadására használhatjuk. Mindazonáltal nem alkalmas magasabb fokú kapcsolatok bizonyos kulcsra vonatkozó megszorításainak megadására, ahogy azt majd a . alfejezetben tárgyaljuk.

A 3.15. ábra a VÁLLALAT adatbázisséma összes szerepkörnevét is ábrázolja.

Kettőnél magasabb fokú kapcsolattípusok

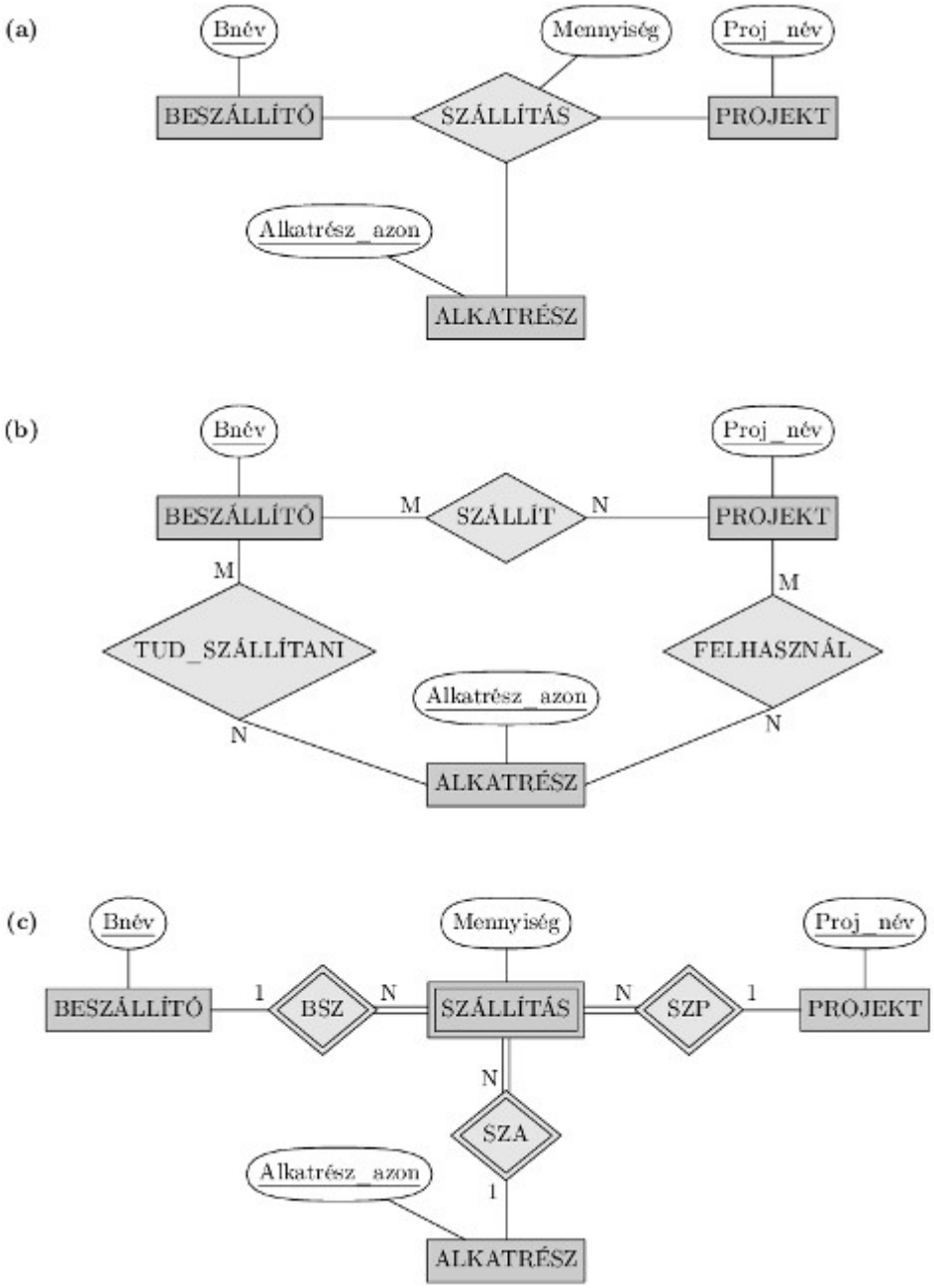
A . alfejezetben a kapcsolattípusok **fokát** a részt vevő egyedtípusok számával definiáltuk, valamint a másodfokú kapcsolattípusokat *binárisnak*, a harmadfokúakat pedig *ternárisnak* neveztük. Ebben az alfejezetben a bináris és a magasabb fokú kapcsolatok közötti különbségeket (például hogy mikor válasszunk magasabb fokú és mikor bináris kapcsolatokat), illetve a magasabb fokú kapcsolatokra vonatkozó megszorításokat vesszük górcső alá.

Választás a bináris és ternáris (vagy magasabb fokú) kapcsolatok között

A 3.16. (a) ábrán egy ternáris kapcsolattípus ER diagram jelölését láthatjuk; ez az ábra a

SZÁLLÍTÁS kapcsolattípus sémáját mutatja, amelyet példány szinten a 3.10. ábra ábrázol. Ne feledjük, hogy a SZÁLLÍTÁS kapcsolathalmaz olyan (s, j, p) kapcsolat-előfordulások halmaza, ahol s egy BESZÁLLÍTÓ, aki szállít egy p ALKATRÉSZ-t a j PROJEKT-hez. Általánosságban egy n-edfokú R kapcsolattípusból n vonal indul ki egy ER diagramban, amelyek R-et kötik össze az egyes részt vevő egyedtípusokkal.

3.16. ábra - Harmadfokú kapcsolattípus. (a) A SZÁLLÍTÁS kapcsolattípus. (b) Három bináris kapcsolattípus, amely nem ekvivalens a SZÁLLÍTÁS-sal. (c) A gyenge egyedtípusként reprezentált SZÁLLÍTÁS.



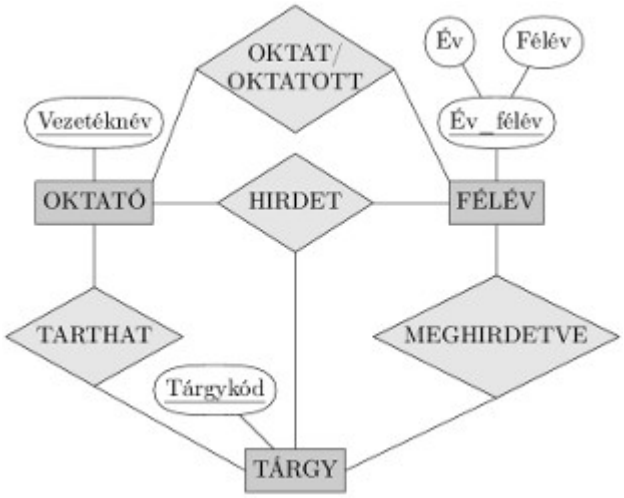
A 3.16. (b) ábra három bináris kapcsolattípus ER diagramját mutatja: TUD_SZÁLLÍTANI, FELHASZNÁL és SZÁLLÍT. Általánosságban egy ternáris kapcsolattípus más információt reprezentál, mint három bináris kapcsolattípus. Tekintsük például a TUD_SZÁLLÍTANI, FELHASZNÁL és SZÁLLÍT bináris kapcsolattípusokat. Tegyük fel, hogy a BESZÁLLÍTÓ és az ALKATRÉSZ közötti TUD_SZÁLLÍTANI kapcsolat akkor tartalmaz egy (s, p) előfordulást, ha az s szállító *tud szállítani* p alkatrészt (valamely projekthez); a PROJEKT és az ALKATRÉSZ közötti FELHASZNÁL kapcsolat akkor tartalmaz egy (j, p) előfordulást, ha a j projekt felhasználja a p

alkatrészt; a BESZÁLLÍTÓ és a PROJEKT közötti SZÁLLÍT kapcsolat pedig akkor tartalmaz egy (s, j) előfordulást, ha az s szállító szállít valamilyen alkatrészt a j projekthez. A TUD_SZÁLLÍTANI, FELHASZNÁL és SZÁLLÍT kapcsolatokban szereplő (s, p), (j, p) és (s, j) három kapcsolat-előfordulás létezéséből nem feltétlenül következik, hogy a SZÁLLÍTÁS ternáris kapcsolatban létezik egy (s, j, p) előfordulás, mivel *a jelentésük különböző*. Gyakran nem egyszerű eldönteni, hogy egy konkrét kapcsolatot egy n-edfokú kapcsolattípusként, vagy több alacsonyabb fokú kapcsolattípussal célszerű-e reprezentálni. A tervezőnek az éppen reprezentált konkrét szituáció szemantikája vagy jelentése alapján kell ezt a döntést meghoznia. Tipikus megoldás, hogy a ternáris kapcsolathoz *hozzáveszünk* még egyet vagy többet a bináris kapcsolatok közül is, ha azok más jelentéstartalommal bírnak, és ha az alkalmazásnak mindegyikre szüksége van.

Egyes adatbázis-tervezési eszközök az ER modell olyan változatain alapulnak, amelyek csak bináris kapcsolatokat engednek meg. Ebben az esetben a ternáris kapcsolatokat, mint például a SZÁLLÍTÁS, részleges kulcs nélküli gyenge egyedtípusként kell reprezentálni, három azonosító kapcsolattal. A három részt vevő egyedtípus (a BESZÁLLÍTÓ, az ALKATRÉSZ és a PROJEKT) együtt alkotják a tulajdonos egyedtípusokat (lásd a 3.16. (c) ábrát). A 3.16. (c) ábra SZÁLLÍTÁS gyenge egyedtípusának egy egyedét tehát a BESZÁLLÍTÓ, az ALKATRÉSZ és a PROJEKT egyedtípusokból vett három tulajdonos egyedének kombinációja azonosítja.

A ternáris kapcsolatot hagyományos egyedtípussal is reprezentálhatjuk egy mesterséges vagy helyettesítő kulcs bevezetésével. Példánkban egy Szállítás_ID kulcs attribútumot használhatunk a SZÁLLÍTÁS egyedtípushoz, amellyel hagyományos egyedtípussá alakítjuk azt. Ekkor három bináris 1:N kapcsolat kapcsolja a SZÁLLÍTÁS-t a három részt vevő egyedtípushoz.

3.17. ábra - Ternáris vagy bináris kapcsolattípusok — egy másik példa.



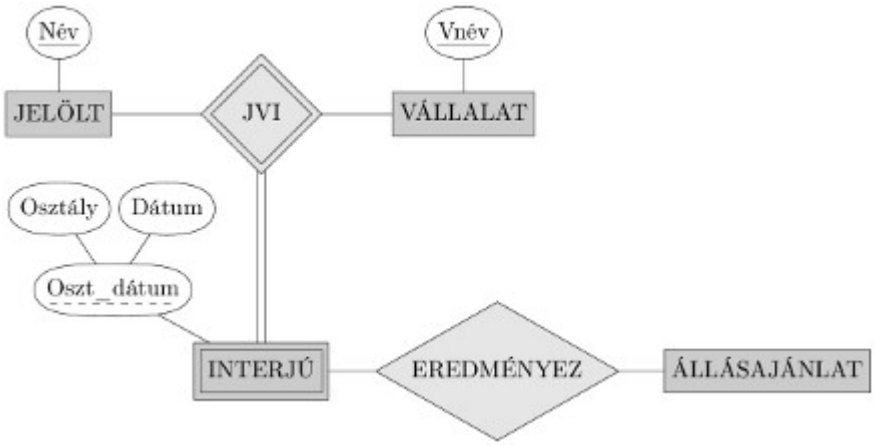
Egy másik példát láthatunk a 3.17. ábrán. A HIRDET ternáris kapcsolattípus olyan oktatókról ad információkat, akik bizonyos félévekben bizonyos tantárgyakat hirdetnek; azaz akkor tartalmaz egy (i, s, c) kapcsolat-előfordulást, ha az i oktató az s félévben meghirdeti a c tantárgyat. A 3.17. ábrán látható három bináris kapcsolattípus jelentése a következő: a TARTHAT egy tantárgyat kapcsol össze azokkal az oktatókkal, akik *tarthatják* az adott tárgyat, az OKTAT/OKTATOTT egy félévet kapcsol össze azokkal az oktatókkal, akik az adott félévben *tartanak/tartottak* órákat, míg a MEGHIRDETVE egy félévet kapcsol össze azokkal a tantárgyakkal, amelyeket az adott félévben hirdetett meg *bármelyik oktató*. Ezek a ternáris és bináris kapcsolatok különböző információkat reprezentálnak, de a kapcsolatok között bizonyos megszorításoknak kell fennállniuk. Például egy (i, s, c) kapcsolat-előfordulás csak akkor szerepelhet a HIRDET kapcsolatban, ha egy (i, s) előfordulás szerepel az OKTAT/OKTATOTT kapcsolatban, egy (s, c) előfordulás szerepel a MEGHIRDETVE kapcsolatban, valamint egy (i, c) előfordulás szerepel a TARTHAT kapcsolatban. A fordítottja azonban nem mindig igaz; előfordulhat, hogy van egy (i, s), egy (s, c) és egy (i, c) előfordulásunk a három bináris kapcsolattípusban, de nincs ezeknek megfelelő (i, s, c) előfordulás a HIRDET

kapcsolatban. Vegyük észre, hogy ebben a példában a kapcsolatok jelentése alapján a HIRDET előfordulásaiból következtethetünk az OKTAT/OKTATOTT és a MEGHIRDETVE előforulásaira, de nem következtethetünk a TARTHAT előfordulásaira; emiatt az OKTAT/OKTATOTT és a MEGHIRDETVE redundánsak és elhagyhatók.

Bár általában három bináris kapcsolat *nem* helyettesíthet egy ternáris kapcsolatot, bizonyos *további megszorítások* fennállása esetén ezt mégis megtehetik. Ha a példánkban a TARTHAT kapcsolat 1:1 számosságú (egy oktató egy tárgyat tarthat, és egy tárgyat csak egy oktató tarthat), akkor a HIRDET ternáris kapcsolat elhagyható, mert kikövetkeztethető a TARTHAT, az OKTAT/OKTATOTT és a MEGHIRDETVE bináris kapcsolatokból. A sématervezőnek minden konkrét szituáció jelentését elemeznie kell ahhoz, hogy el tudja dönteni, hogy mely bináris és ternáris kapcsolattípusok szükségesek.

Figyeljük meg, hogy egy gyenge egyedtípusnak lehet ternáris (vagy *n*-edfokú) azonosító kapcsolattípusa. Ebben az esetben a gyenge egyedtípusnak *több* tulajdonos egyedtípusa lehet. Egy példa látható a 3.18. ábrán.

3.18. ábra - Az INTERJÚ gyenge egyedtípus ternáris azonosító kapcsolattípussal.



A ternáris (vagy magasabb fokú) kapcsolatok megszorításai

Kétféle jelölés létezik az *n*-edfokú kapcsolatok strukturális megszorításainak megadására, és ezek különböző megszorításokat definiálnak. Ezért *mindkettőt használni kell*, ha fontos számunkra hogy teljesen megadjuk egy ternáris vagy magasabb fokú kapcsolat strukturális megszorításait. Az első jelölés a bináris kapcsolatok számosságának 3.2. ábrán látható jelölésén alapul. Itt egy 1-est, egy M-et vagy egy N-et írunk mindegyik részvételt jelző vonal mellé (az M és az N szimbólumok mindegyike *sok* vagy *bármennyi* jelentésű).[13] Ezt a megszorítást a 3.16. ábra SZÁLLÍTÁS kapcsolatának segítségével illusztráljuk.

Emlékezhetünk rá, hogy a SZÁLLÍTÁS kapcsolathalmaz olyan (s, j, p) kapcsolat-előfordulások halmaza, ahol s egy BESZÁLLÍTÓ, j egy PROJEKT, p pedig egy alkatrész. Tegyük fel, hogy van egy olyan megszorításunk, amely szerint egy adott projekt-alkatrész kombináció esetén csak egy beszállítót használunk (csak egy beszállító szállít egy adott alkatrészt egy adott projekthez). Ebben az esetben 1-est írunk a BESZÁLLÍTÓ részvételéhez és M-et, illetve N-et a PROJEKT és az ALKATRÉSZ részvételéhez a 3.16. ábrán. Ez azt a megszorítást definiálja, hogy egy adott (j, p) kombináció legfeljebb egyszer jelenhet meg a kapcsolathalmazban, mivel minden ilyen (PROJEKT, ALKATRÉSZ) kombináció egyértelműen meghatároz egy konkrét beszállítót. Így minden (s, j, p) kapcsolat-előfordulást egyértelműen azonosít a kapcsolathalmazban a (j, p) kombinációja, amely (j, p)-t a kapcsolathalmaz kulcsává teszi. Ebben a jelölésben azok a részvételek, amelyek mellett 1-es szerepel, nem szükségesek a kapcsolathalmaz azonosító kulcsához.[14]

A második jelölés a bináris kapcsolatokra vonatkozó, 3.15. ábrán látható (min, max) jelölésen alapul. Egy részvétel mellé írt (min, max) itt azt definiálja, hogy minden egyed legalább *min*, legfeljebb *max* kapcsolat-előforduláshoz kötődik a kapcsolathalmazban. Az ilyen megszorítások nem képezhetik az alapját egy *n*-edfokú kapcsolat kulcsa meghatározásának, ahol $n > 2$,[15] ehelyett egy másfajta megszorítást definiálnak, amely arra vonatkozóan ad korlátozást, hogy hány kapcsolat-előfordulásban vehetnek részt az egyes egyedek.

Összefoglalás

Ebben a fejezetben egy magas szintű koncepcionális adatmodell, az egyed-kapcsolat (ER) modell modellezési fogalmait ismertettük. A magas szintű adatmodelleknek az adatbázis-tervezési folyamatban játszott szerepével kezdtük a tárgyalást, majd bemutattuk a VÁLLALAT adatbázishoz tartozó adatbázis követelményeket, amely egyike a jegyzeten keresztülívelő példáknak. Definiáltunk olyan alapvető ER modellbeli fogalmakat, mint az egyedek és az attribútumaik. Ezután beszéltünk a NULL értékekről, és bemutattuk az attribútumok különböző típusait, amelyeket tetszőlegesen egymásba ágyazhatunk, hogy komplex attribútumokat képezzünk:

- Egyszerű vagy atomi
- Összetett
- Többértékű

Röviden érintettük a tárolt és származtatott attribútumok témakörét is. Ezután a séma vagy „intenzió” szintű ER modellbeli fogalmakat ismertettük:

- Egyedtípusok és a nekik megfelelő egyedhalmazok
- Az egyedtípusok kulcs attribútumai
- Az attribútumok értékkészlete (tartományai)
- Kapcsolattípusok és a nekik megfelelő kapcsolathalmazok
- Az egyedtípusok részvételi szerepkörei a kapcsolattípusokban

Bemutattunk két módszert a kapcsolattípusok strukturális megszorításainak a megadására. Az első módszer a strukturális megszorítások két típusát különbözteti meg:

- Számosság (1:1, 1:N, M:N bináris kapcsolatok esetén)
- Részvételi megszorítások (totális, részleges)

Megjegyeztük, hogy a strukturális megszorítások megadásának a másik módja az, hogy (min, max) alakú minimum és maximum értékeket adunk meg a kapcsolattípusban részt vevő minden egyedtípus részvételéhez. Ismertettük a gyenge egyedtípusokat és az olyan hozzájuk kapcsolódó fogalmakat, mint a tulajdonos egyedtípusok, azonosító kapcsolattípusok és részleges kulcs attribútumok.

Az egyed-kapcsolat sémákat grafikusan ER diagramokkal reprezentálhatjuk. Megmutattuk, hogyan tervezhetünk egy ER sémát a VÁLLALAT adatbázishoz úgy, hogy először az egyedtípusokat és azok attribútumait definiáljuk, majd finomítjuk a tervet kapcsolattípusok hozzáadásával. Bemutattuk a VÁLLALAT adatbázisséma ER diagramját. Részletesen foglalkoztunk a ternáris és magasabb fokú kapcsolattípusokkal, és vázoltuk azokat a körülményeket, amelyek között eltérnek a bináris kapcsolatoktól.

Az eddig bemutatott ER modellbeli fogalmak — egyedtípusok, kapcsolattípusok, attribútumok, kulcsok és strukturális megszorítások — alkalmasak hagyományos üzleti adatfeldolgozó adatbázis alkalmazások modellezésére. Sok újabb, komplexebb alkalmazáshoz — mint például mérnöki tervezés, orvosi információs rendszerek vagy telekommunikáció — azonban további fogalmakra

van szükségünk, ha nagyobb pontossággal szeretnénk őket modellezni. Néhány haladó modellezési fogalommal foglalkozunk a 4. fejezetben.

Áttekintő kérdések

1. Ismertesse a magas szintű (koncepcionális) adatmodell szerepét az adatbázistervezési folyamatban!
2. Sorolja fel azokat az eseteket, amelyeknél előfordulhat a NULL érték használata!
3. Definiálja a következő fogalmakat: *egyed*, *attribútum*, *attribútumérték*, *kapcsolatelőfordulás*, *összetett attribútum*, *többértékű attribútum*, *származtatott attribútum*, *komplex attribútum*, *kulcs attribútum* és *értékhalmaz* (*tartomány*).
4. Mi az egyedtípus? Mi az egyedhalmaz? Fejtse ki, hogy mi a különbség az egyed, az egyedtípus és az egyedhalmaz között!
5. Fejtse ki, hogy mi a különbség az attribútum és az értékhalmaz között!
6. Mi a kapcsolattípus? Fejtse ki, hogy mi a különbség a kapcsolatelőfordulás, a kapcsolattípus és a kapcsolathalmaz között!
7. Mi a részvételi szerepkör? Mikor szükséges szerepkörneveket használni a kapcsolattípusok leírásánál?
8. Ismertesse a kapcsolattípusokra előírható strukturális megszorítások megadásának két lehetséges módját! Mik az előnyei és a hátrányai ezeknek?
9. Milyen feltételek mellett lehet egy bináris kapcsolattípus egy attribútumát az egyik résztvevő egyedtípus egy attribútumává átalakítani?
10. Ha a kapcsolatokra mint attribútumokra gondolunk, mik lesznek ezen attribútumok értékhalmazai? Az adatmodellek melyik osztálya alapul ezen a fogalmon?
11. Mit értünk rekurzív kapcsolattípus alatt? Adjon néhány példát rekurzív kapcsolattípusra!
12. Mikor használjuk a gyenge egyedtípus fogalmát az adatmodellezésben? Definiálja a *tulajdonos egyedtípus*, a *gyenge egyedtípus*, az *azonosító kapcsolattípus* és a *részleges kulcs* fogalmát!
13. Lehet-e egy gyenge egyedtípus azonosító kapcsolatának a fokszáma kettőnél nagyobb? Illusztrálja válaszát példákkal!
14. Ismertesse az ER séma ER diagramként történő ábrázolásának konvencióit!
15. Ismertesse az ER séma diagramoknál használatos elnevezési szokásokat!

Feladatok

1. Tekintse a hallgatók tanulmányi eredményeinek nyilvántartására használt EGYETEM adatbázis követelményeinek alábbi listáját! Ez hasonló az 1.2. ábrán látható adatbázishoz, de nem azonos azzal.
    a. Az egyetem nyilvántartja a hallgatók nevét, Neptun-kódját, személyi számát, tartózkodási helyét, állandó lakcímét, telefonszámát, születési dátumát, nemét, évfolyamát (1-től 5-ig), szakját, szakirányát (ha van) és képzési szintjét (BSc, MSc, PhD). Egyes felhasználói alkalmazásoknak szükségük van a hallgató állandó lakcíméből az irányítószámra, a megyére és a városra, valamint a hallgató vezetéknevére. Minden egyes hallgatóhoz egyedi személyi szám és Neptun-kód tartozik.

b. Egy kart a neve, a kódja, a dékáni hivatalának az iroda- és telefonszáma, valamint az intézménye (amihez tartozik) jellemez. Minden egyes kar egyedi névvel és kóddal rendelkezik.

c. A tantárgyak névvel, leírással, tantárgykóddal, félévenkénti óraszámmal, szinttel és egy őket hirdető tanszékkel rendelkeznek. Minden tartárgy tantárgykódja egyedi.

d. Minden kurzushoz tartozik egy oktató, egy félév, egy tanév, egy tantárgy és egy kurzuskód. A kurzuskód különbözteti meg egy tantárgy ugyanazon félévben/tanévben tartott kurzusait; értékei 1, 2, 3, … lehetnek, az egyes félévekben tartott kurzusok számával bezárólag.

e. A vizsgaeredmény azt mondja meg, hogy egy hallgató egy kurzust milyen érdemjeggyel teljesített (1-től 5-ig).

Tervezzen meg egy ER sémát ehhez az alkalmazáshoz, és rajzolja meg a sémához tartozó ER diagramot! Adja meg az összes egyedtípus kulcs attribútumait, és a kapcsolattípusok strukturális megszorításait! Jegyezze fel a nem specikifált követelményeket, és adja meg a megfelelő feltételezéseket, hogy a specifikáció teljes legyen!

2. Az összetett és többértékű attribútumokat tetszőleges szintig egymásba ágyazhatjuk. Tegyük fel, hogy a HALLGATÓ egyedtípushoz szeretnénk egy olyan attribútumot tervezni, amely a korábbi felsőfokú tanulmányait tartja nyilván! Egy ilyen attribútum minden korábban látogatott felsőoktatási intézményhez tartalmaz egy bejegyzést, és minden bejegyzés az intézmény nevéből, egy kezdő és egy befejező dátumból, diploma bejegyzésekből (az adott intézmény által kibocsátott diplomákról, ha vannak) és tantárgy bejegyzésekből (az adott inzézményben teljesített tantárgyakról) áll. Az egyes diploma bejegyzések a diploma nevét, illetve a diploma kiadásának évét és hónapját tartalmazzák, míg a tantárgy bejegyzések egy tantárgynévből, egy tanévből, egy félévből és egy jegyből állnak. Tervezzen egy attribútumot, amely mindezeket az információkat tárolja! Használja a 3.5. ábrán látható jelöléseket!

3. Készítsen egy alternatív tervet a 3.2 feladatban leírt attribútumhoz úgy, hogy csak egyedtípusokat (beleértve a gyenge egyedtípusokat is, ha szükségesek) és kapcsolattípusokat használ!

4. Tekintse a 3.19. ábrán látható ER diagramot, amely egy légitársaság helyfoglalási rendszerének egy egyszerűsített sémáját ábrázolja! Gyűjtse össze az ER diagramból azokat a követelményeket és megszorításokat, amelyek ezt a sémát eredményezték! Próbáljon meg a lehető legprecízebb lenni a követelmények és megszorítások specifikálásakor!

3.19. ábra - ER diagram a REPÜLŐGÉP-MENETREND adatbázissémához.

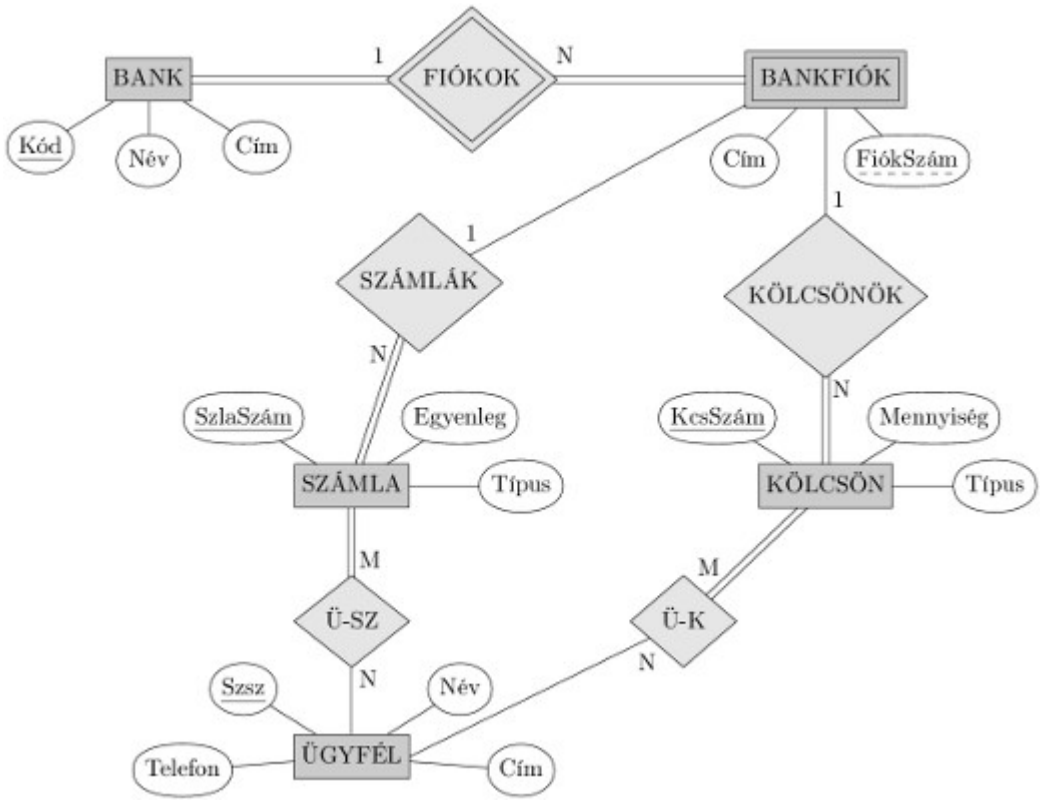
5. Az 1. és 2. fejezetben az adatbázis környezetéről és az adatbázis felhasználóiról tárgyaltunk. Egy ilyen környezet leírásához sok egyedtípust használhatunk fel, mint például a DBMS, a tárolt adatbázis, a DBA és a katalógus/adatszótár. Próbálja meg megadni az összes olyan egyedtípust, amelyek teljesen leírnak egy adatbázisrendszert és a környezetét; majd adja meg a közöttük lévő kapcsolattípusokat, és rajzolja fel azt az ER diagramot, amely egy ilyen általános adatbázis-környezetet ír le!

6. Tervezzen egy ER sémát az Egyesült Államok képviselőházában az aktuális kétéves kongresszusi ciklusban leadott szavazatokat leíró információk nyilvántartására. Az adatbázisnak tárolnia kell az összes amerikai ÁLLAM Nevét (pl. 'Texas', 'New York', 'Kalifornia'), és tartalmaznia kell az adott állam Régióját (amelynek tartománya {'Északkelet', 'Középnyugat', 'Délkelet', 'Délnyugat', 'Nyugat'}). A képviselőház minden

KÉPVISELŐ-jét a Nevével, az általa képviselt Körzettel, a Kezdő_dátummal, amikor a képviselőt először megválasztották, valamint azzal a politikai Párttal írjuk le, amelyikhez tartozik (ennek tartománya {'republikánus', 'demokrata', 'független', 'egyéb'}). Az adatbázis nyilvántartja az összes ELŐTERJESZTÉS-t (azaz benyújtott törvényjavaslatot), beleértve az Előterjesztés_címét, az előterjesztés Szavazási_dátumát, hogy az előterjesztés Átment_vagy_megbukott (ennek tartománya {'igen', 'nem'}) és a Szponzort (azokat a képviselőket, akik szponzorálták — azaz támogatták — az előterjesztést). Az adatbázis nyilvántartja azt is, hogy hogyan szavaztak az egyes képviselők az egyes előterjesztésekre (a Szavazat attribútum tartománya {'igen', 'nem', 'tartózkodás', 'távollét'}). Rajzoljon fel egy ER sémna diagramot ehhez az alkalmazáshoz! Írja fel világosan az esetlegesen megfogalmazódó feltételezéseit!

7. Egy olyan adatbázist készítünk, amely egy sportliga csapatait és mérkőzéseit tartja nyilván. A csapatok néhány játékossal rendelkeznek, akik közül nem mindenki vesz részt minden mérkőzésen. Nyilván kell tartanunk, hogy az egyes csapatokból az egyes mérkőzéseken mely játékosok vettek részt, milyen pozícióban játszottak a meccsen, és hogy mi lett a mérkőzés végeredménye. Tervezzen egy ER séma diagramot ehhez az alkalmazáshoz, és írja le az esetlegesen megfogalmazódó feltételezéseit! Válassza ki a kedvenc sportágát (pl. futball, vízilabda, kézilabda, jégkorong)!
8. Tekintse a BANK adatbázis egy részének a 3.20. ábrán látható ER diagramját! Minden banknak több bankfiókja lehet, és minden bankfiók több számlát és kölcsönt kezelhet.

3.20. ábra - ER diagram a BANK adatbázissémához.



a. Sorolja fel az ER diagramon látható (nem gyenge) egyedtípusokat!
b. Van gyenge egyedtípus? Ha igen, adja meg a nevét, a részleges kulcsát és az azonosító kapcsolattípusát!
c. Milyen megszorításokat definiál az ábrán látható gyenge egyedtípus részleges kulcsa és azonosító kapcsolattípusa?
d. Sorolja fel a kapcsolattípusok neveit, és határozza meg az egyes kapcsolattípusokban

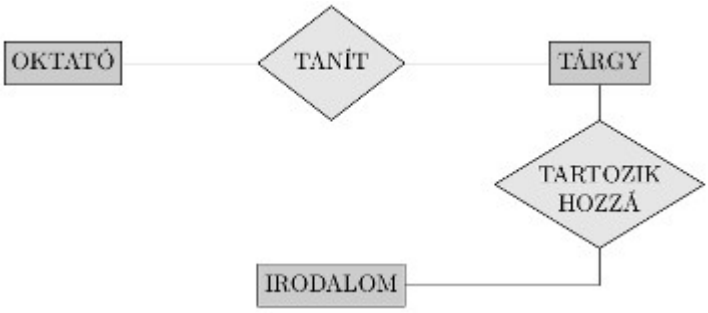
részt vevő egyedtípusok (min, max) megszorításait! Igazolja állításait!

e. Sorolja fel tömören azokat a felhasználói követelményeket, amelyek ehhez az ER sématervhez vezettek!

f. Tegyük fel, hogy minden ügyfélnek legalább egy számlával kell rendelkeznie, de egy időben legfeljebb csak két kölcsöne lehet, illetve hogy egy bankfiók nem kezelhet 1000-nél több kölcsönt! Hogyan mutatkoznak meg ezek a előírások a (min, max) megszorításokban?

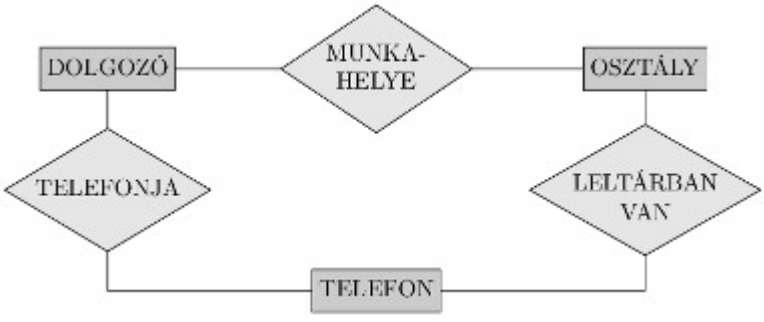
9. Tekintsük a 3.21. ábrán látható ER diagramot! Tegyük fel, hogy egy dolgozó legfeljebb két osztályon dolgozhat, vagy egyetlenegy osztályhoz sem tartozik! Tegyük fel, hogy minden osztály legalább egy és legfeljebb három telefonszámmal rendelkezik! Adja meg a (min, max) megszorításokat a diagramon! *Írja fel világosan az esetlegesen megfogalmazódó további feltételezéseit!* Milyen feltételek mellett lenne a TELEFONJA kapcsolat redundáns ebben a példában?

3.21. ábra - A VÁLLALAT adatbázis ER diagramjának egy részlete.



10. Tekintsük a 3.22. ábrán látható ER diagramot! Tegyük fel, hogy egy tantárgyhoz tartozhat is ajánlott irodalom, meg nem is, de az irodalom definíció szerint olyan könyv, amely valamely tantárgyhoz tartozik! Egy tantárgyhoz nem tartozhat ötnél több könyv. Az oktatók legalább kettő és legfeljebb négy tantárgyat oktatnak. Adja meg a (min, max) megszorításokat a diagramon! *Írja fel világosan az esetlegesen megfogalmazódó további feltételezéseit!* Ha felveszünk egy FELHASZNÁL kapcsolatot az OKTATÓ és az IRODALOM közé, milyen (min, max) megszorításokat írna mellé? Miért?

3.22. ábra - A TANTÁRGYAK adatbázis ER diagramjának egy részlete.



11. Tekintsünk egy KURZUS egyedtípust az EGYETEM adatbázisban, amely a tantárgyak kurzushirdetéseit írja le! A KURZUS attribútumai a Kurzuskód, a Tanév, a Félév, a Tantárgykód, az Oktató, az Épület (ahol a kurzust tartják), a Tanterem (ahol a kurzust tartják), a Napok (amelynek a tartománya a hét napjainak a lehetséges kombinációi, amikor a kurzust tarthatják: {'HSzP', 'HSz', 'KCs' stb.}) és az Órák (amelynek tartománya az összes lehetséges időintervallum, amikor a kurzust tarthatják: {9:00–10:00, 10:00–11:00, …, 15:30–17:00, 17:30–18:30 stb.}). Tételezzük fel, hogy a Kurzuskód minden kurzusnál egyedi egy konkrét tanév/félév kombináció esetén (azaz ha egy tantárgyat egy adott

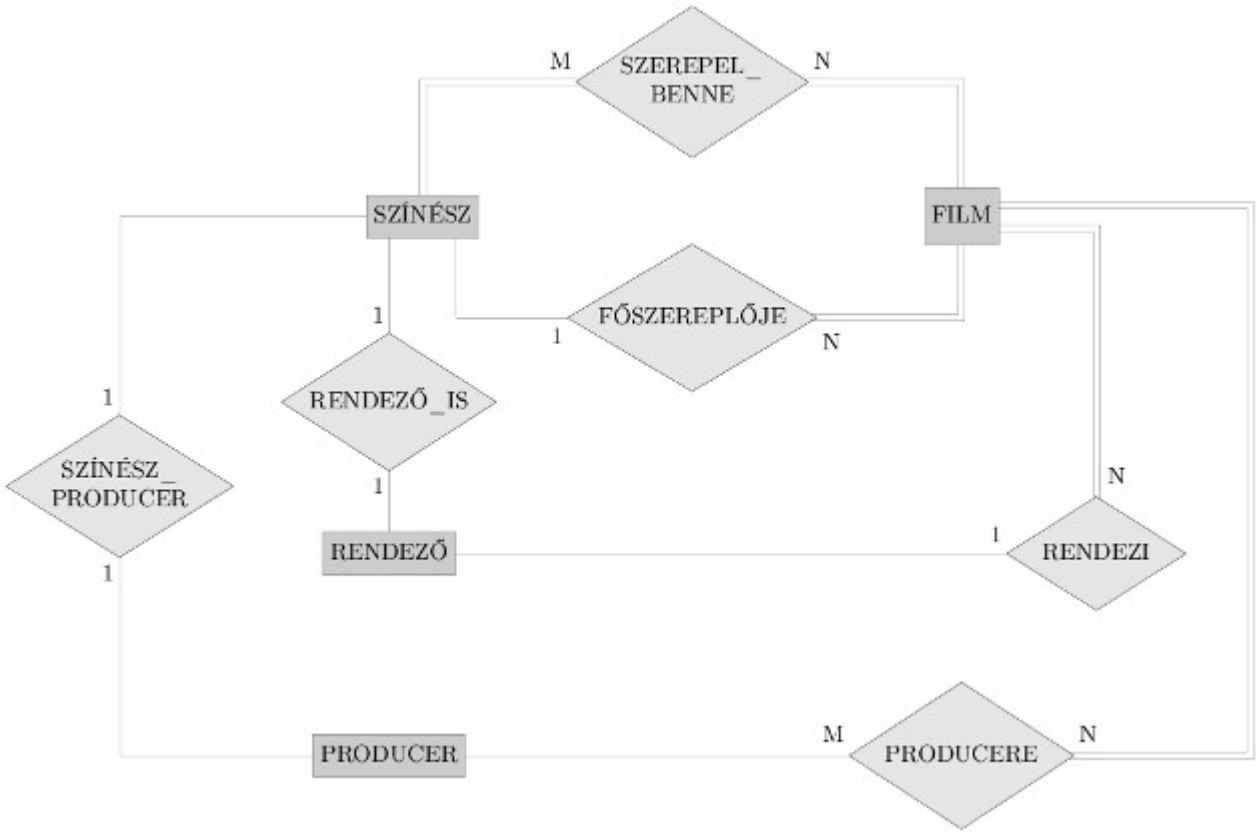
félévben többször is meghirdetnek, akkor a kurzusait 1-től kezdődően sorszámozzuk (1, 2, 3 stb.)! A kurzusnak számos összetett kulcsa van, és bizonyos attribútumok egynél több kulcsban is szerepelnek. Azonosítson be három összetett kulcsot, és mutassa meg, hogyan reprezentálhatjuk őket egy ER séma diagramon!

12. A számosságok gyakran megszabják egy adatbázis részletes tervezésének a folyamatát. A számosság a kérdéses egyedtípusok valós világbeli jelentésétől függ, és a konkrét alkalmazás definiálja. Javasoljon számosságokat az alábbi bináris kapcsolatokhoz az egyedtípusok hétköznapi jelentése alapján! Írja fel világosan az esetlegesen megfogalmazódó feltételezéseit!

	Egyik egyed	Számosság	Másik egyed
1.	HALLGATÓ		SZEMÉLYI_IGAZOLVÁNY
2.	HALLGATÓ		OKTATÓ
3.	TANTEREM		FAL
4.	ORSZÁG		AKTUÁLIS_ELNÖK
5.	TANTÁRGY		IRODALOM
6.	TÉTEL (amely egy rendelésen található)		RENDELÉS
7.	HALLGATÓ		ÉVFOLYAM
8.	ÉVFOLYAM		OKTATÓ
9.	OKTATÓ		IRODA
10.	EBAY_ÁRVERÉSI_TÉTEL		EBAY_LICIT

13. Tekintse a FILMEK adatbázis 3.23. ábrán látható ER sémáját! Tegyük fel, hogy a FILMEK egy feltöltött adatbázis! A színész egy általános kifejezés, amelybe a színésznőket is beleértjük. Az ER sémán látható megszorításokat figyelembe véve válaszoljon az alábbi állításokra az *igaz*, *hamis* vagy *talán* szavak valamelyikével. Azon állításokra adjon *talán* választ, amelyekről bár explicit módon nem látható, hogy igazak, de a séma alapján azt sem lehet bebizonyítani róluk, hogy hamisak. Igazolja minden válaszát!
    a. Nincs olyan színész az adatbázisban, aki egyetlen filmben sem szerepel.
    b. Van néhány olyan színész, aki tíznél több filmben játszik.
    c. Egyes színészek több filmben is főszerepet játszanak.
    d. Egy filmnek legfeljebb két főszereplője lehet.
    e. Minden rendező szerepel színészként valamelyik filmben.
    f. Egyik producer sem színész.
    g. Egy producer nem lehet színész egy másik filmben.
    h. Vannak olyan filmek, amelyekben több mint egy tucat színész szerepel.
    i. Egyes producerek rendezők is.
    j. A legtöbb filmnek egy rendezője és egy producere van.
    k. Egyes filmeknek egy rendezője, de több producere van.
    l. Vannak olyan színészek, akik főszerepet játszanak egy filmben, rendezői egy filmnek, valamint producerei valamely filmnek.
    m. Egyik film rendezője sem szerepel színészként abban a filmben.

3.23. ábra - ER diagram a FILMEK adatbázissémához.



14. A FILMEK adatbázis 3.23. ábrán látható ER sémája mellé rajzoljon fel egy példánydiagramot három nemrég vetített filmmel! Rajzoljon példányokat az összes egyedtípushoz: legyenek benne filmek, színészek, producerek és rendezők; a kapcsolatok példányait úgy hozza létre, ahogyan az adott filmek esetén a valóságban léteznek!
15. Rajzolja fel a 3.1 feladat UML diagramját! Az UML tervnek az alábbi követelményeknek kell megfelelnie:
    a. A hallgatóknak képesnek kell lenniük kiszámítani a tanulmányi átlagukat, illetve felvenni vagy leadni főszakokat és minorokat.
    b. A tanszékeknek képesnek kell lenniük új tantárgyakat felvenni és létezőeket törölni, valamint új oktatót felvenni és meglévőt elbocsátani.
    c. Az oktatóknak képesnek kell lenniük egy hallgatónak egy adott tantárgyból szerzett jegyét felvinni és módosítani.

    Megjegyzés: A felsorolt tevékenységek közül néhány több osztályt is érinthet.

[2] Irányítószámnak Magyarországon a négyjegyű postai kódot nevezzük.

[3] Az XML-t jól ismerők számára megjegyezzük, hogy a komplex attribútumok hasonlóak az XML komplex elemeihez.

[4] Egy V halmaz $P(V)$ **hatványhalmaza** V összes részhalmazának a halmaza.

[5] Az **egyelemű halmaz** olyan halmaz, amelynek csak egyetlen eleme van.

[6] A kapcsolattípusok attribútumként való reprezentálása a **funkcionális adatmodellek** osztályában használatos. Az objektumorientált adatbázisokban a kapcsolatok referencia attribútumokként reprezentálhatók, vagy csak egy irányban, vagy mindkét irányban, egymás inverzeként. Relációs

adatbázisokban (lásd 5. fejezet) a külső kulcs a referencia attribútumok egy fajtája, amelyet a kapcsolatok reprezentálására használunk.

[7] Az N azt jelenti, hogy *tetszőleges számú* (0 vagy több) kapcsolódó egyed létezhet.

[8] Az azonosító egyedtípust néha **szülő egyedtípusnak** vagy **domináns egyedtípusnak** is nevezik.

[9] A gyenge egyedtípust néha **gyermek egyedtípusnak** vagy **alárendelt egyedtípusnak** is nevezik.

[10] A részleges kulcsot néha **diszkriminátornak** is nevezik.

[11] A minivilágbeli szabályokat, amelyek meghatározzák a megszorításokat, néha *üzleti szabályoknak* is nevezzük, mivel őket az *üzlet* vagy szervezet határozza meg, amely használni fogja majd az adatbázist.

[12] Bizonyos jelölésrendszerek — különösen azok, amelyeket az objektummodellezési módszertanok (mint például az UML) használnak — a (min, max) jelölést az általunk mutatotthoz képest *az ellenkező oldalra* írják. A 3.15. ábra MUNKAHELYE kapcsolata esetén például az (1, 1) szerepelne az OSZTÁLY oldalán, míg a (4, N) a DOLGOZÓ oldalán. Mi itt Abrial (1974) eredeti jelölését használtuk.

[13] Ez a jelölés lehetővé teszi számunkra a kapcsoló reláció kulcsának meghatározását, ahogyan a 7. fejezetben tárgyaljuk.

[14] Ez a bináris kapcsolatok számosságára is igaz.

[15] A (min, max) megszorítások a bináris kapcsolatok esetén azonban meghatározhatják a kulcsokat.

4. fejezet - A kibővített egyed-kapcsolat (Enhanced ER, EER) modell

Tartalom

Az ER modellezés 3. fejezetben tárgyalt fogalmai számos — főleg üzleti és ipari adatfeldolgozó alkalmazásokat magában foglaló — *tradicionális* adatbázis alkalmazás számára készített adatbázisséma reprezentálásához elegendőek. Az 1970-es évek végétől kezdve azonban az adatbázis alkalmazások tervezői megpróbáltak pontosabb adatbázissémákat tervezni, amelyek precízebben fejezik ki az adatok tulajdonságait és a megszorításokat. Ez különösen fontos volt az adatbázis

technológia olyan újabb alkalmazásai számára, mint például az adatbázisokat használó számítógéppel segített tervezés és gyártás (CAD/CAM)[16], a telekommunikáció, az összetett szoftverrendszerek, valamint a térinformatikai rendszerek (GIS), sok más egyéb alkalmazás mellett. Az ilyen típusú adatbázisok sokkal összetettebb követelményekkel bírnak, mint a tradicionálisabb alkalmazások. Ez további *szemantikus adatmodellezési* fogalmak kialakulásához vezetett, amelyeket beolvasztottak az olyan koncepcionális adatmodellekbe, mint például az ER modell. A szakirodalomban különféle szemantikus adatmodelleket ajánlanak. Ezen fogalmak közül többet a számítástudomány kapcsolódó területein is hasznosítanak, mint például az **ismeretreprezentációt** a mesterséges intelligencia, vagy az **objektum alapú modellezést** a szoftverfejlesztés területén.

Ebben a fejezetben olyan lehetőségeket mutatunk be, amelyeket a szemantikus adatmodellekhez javasoltak, és megmutatjuk, hogyan lehet az ER modellt kiterjeszteni úgy, hogy magában foglalja ezeket a fogalmakat, és így megkapjuk a **kibővített ER** (**EER**) modellt.[17] A . alfejezetben azzal kezdünk, hogy bevezetjük az *osztály/alosztály kapcsolat* és a *típusöröklődés* fogalmakat az ER modellbe. Ezután a . alfejezetben hozzávesszük ezekhez a *specializáció* és a *generalizáció* fogalmakat. A . alfejezet a specializációra/generalizációra megadható *megszorítások* különböző típusait tárgyalja, a . alfejezet pedig bemutatja, hogyan lehet modellezni az unió konstrukciót a *kategória* fogalmának bevezetésével az EER modellbe. A . alfejezet az EGYETEM adatbázissémát mutatja be az EER modellben, és összefoglalja az EER modellbeli fogalmakat formális definíciók segítségével.

A . alfejezet összefoglalja a fejezetet.

Ha az olvasó a koncepcionális modellezést részletesebben szeretné megismerni, akkor a 4. fejezetet a 3. folytatásaként célszerű áttekinteni. Ha azonban az ER modellezést csak alapszinten szeretné megismerni, akkor ez a fejezet kihagyható. Egy másik lehetőség, hogy az olvasó ennek a fejezetnek csak az utolsó alfejezeteit ugorja át (a . alfejezettől a . alfejezetig).

Alosztályok, szuperosztályok és öröklődés

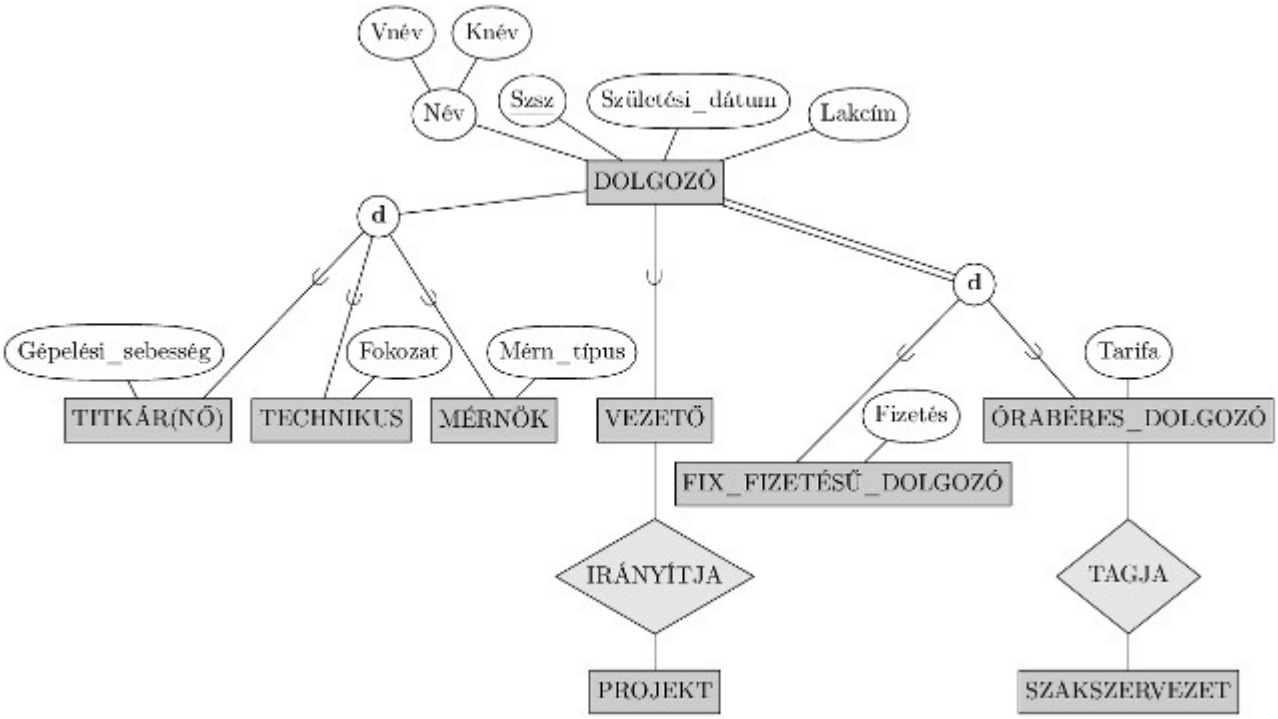
Az EER modell az ER modell összes modellezési eszközét tartalmazza, ezeket a 3. fejezetben mutattuk be. Ezeken felül tartalmazza még az **alosztály** és **szuperosztály** fogalmakat, valamint a **specializáció** és **generalizáció** kapcsolódó fogalmakat is (lásd a . és . alfejezeteket). Egy további EER-beli eszköz a **kategória** vagy **unió típus** (lásd a . alfejezetet), amelyet objektumok egy olyan kollekciójának a reprezentációjára használunk, amely különböző egyedtípusok objektumainak az *uniója*. Ezekhez a fogalmakhoz kötődik az **attribútum- és kapcsolatöröklődés** fontos mechanizmusa. Sajnos ezekhez a fogalmakhoz nem létezik szabványos terminológia, ezért mi a legelterjedtebb terminológiát fogjuk használni. Az alternatív elnevezéseket lábjegyzetekben adjuk meg. Egy grafikus jelölésrendszert is megadunk az EER sémákon felbukkanó sémaelemek megjelenítésére. Az eredményül kapott sémadiagramot **kibővített ER** vagy **EER diagramnak** nevezzük.

Az első EER modellbeli fogalom, amelyet tárgyalunk, az egyedtípus **alosztályának** fogalma. Ahogyan a 3. fejezetben tárgyaltuk, az egyedtípust használjuk mind *az egyed típusának*, mind *az adott típus* (adatbázisban létező) *egyedei kollekciójának* vagy *egyedhalmazának* reprezentálására. A DOLGOZÓ egyedtípus például leírja az egyes dolgozó egyedek típusát (azaz az attribútumait és a kapcsolatait), de egyúttal a VÁLLALAT adatbázisban található DOLGOZÓ egyedek aktuális halmazára is utal. Sok esetben egy egyedtípus egyedeinek számos jelentéssel bíró alcsoportja létezik, amelyeket explicit módon kell reprezentálni az adatbázis alkalmazásban játszott fontos szerepük miatt. Azokat az egyedeket például, amelyek tagjai a DOLGOZÓ egyedtípusnak, tovább csoportosíthatjuk a TITKÁR(NŐ), a MÉRNÖK, a VEZETŐ, a TECHNIKUS, a FIX_FIZETÉSŰ_DOLGOZÓ, az ÓRABÉRES_DOLGOZÓ stb. csoportokba. Az egyes felsorolt csoportokhoz tartozó egyedhalmazok a DOLGOZÓ egyedtípushoz tartozó egyedek részhalmazai, ami azt jelenti, hogy minden egyed, amely tagja ezen alcsoportok egyikének, az egyben dolgozó is.

Ezeknek az alcsoportoknak mindegyikét a DOLGOZÓ egyedtípus **alosztályának** nevezzük, a DOLGOZÓ egyedtípus pedig ezen alosztályok mindegyikének a **szuperosztálya**. A 4.1. ábra mutatja, hogyan lehet ezeket a fogalmakat grafikusan ábrázolni az EER diagramokon. (A 4.1. ábrán karikába írt betű jelentését a . alfejezetben fogjuk elmagyarázni.)

Egy szuperosztály és az alosztályainak egyike közötti kapcsolatot **szuperosztály/alosztály** vagy egyszerűen **osztály/alosztály** kapcsolatnak[18] nevezzük. Az előző példánkban a DOLGOZÓ/VEZETŐ és a DOLGOZÓ/TECHNIKUS két osztály/alosztály kapcsolat. Fontos megjegyezni, hogy az alosztály egy egyede *ugyanazt a valós világbeli egyedet* reprezentálja, mint a szuperosztály valamely tagja; a 'Nagy Julianna' TITKÁR(NŐ) egyed a 'Nagy Julianna' DOLGOZÓ egyed is egyben. Az alosztály tagja tehát ugyanaz, mint a szuperosztálybeli egyed, de egy eltérő, *speciális szerepkörben*. Amikor azonban egy szuperosztály/alosztály kapcsolatot implementálunk az adatbázisrendszerben, az alosztály egy egyedét különálló adatbázis objektumként reprezentálhatjuk; például egy különálló rekordként, amely a szuperosztálybeli egyedéhez egy kulcs attribútumon keresztül kapcsolódik. A alfejezetben különböző lehetőségeket mutatunk a szuperosztály/alosztály kapcsolatok reprezentálására a relációs adatbázisokban.

4.1. ábra - A DOLGOZÓ három specializációja: {TITKÁR(NŐ), TECHNIKUS, MÉRNÖK}, {VEZETŐ} és {ÓRABÉRES_DOLGOZÓ, FIX_FIZETÉSŰ_DOLGOZÓ}.



Egy egyed nem létezhet az adatbázisban úgy, hogy csupán egy alosztálynak a tagja; a szuperosztálynak is tagja kell, hogy legyen. Egy ilyen egyedet opcionálisan akárhány alosztály tagjaként szerepeltethetünk. Például egy fix fizetésű dolgozó, aki egyúttal mérnök is, a DOLGOZÓ egyedtípus MÉRNÖK és FIX_FIZETÉSŰ_DOLGOZÓ alosztályaihoz is tartozik. Nem szükséges azonban, hogy egy szuperosztály minden egyede valamely alosztálynak is egyede legyen.

Az alosztályokhoz kapcsolódó fontos fogalom a **típusöröklődés**. Idézzük fel, hogy egy egyed típusa az általa birtokolt attribútumokkal és azokkal a kapcsolattípusokkal van definiálva, amelyekben részt vesz. Mivel az alosztály egy egyede ugyanazt a valós világbeli egyedet reprezentálja, mint a szuperosztály egy egyede, ezért a speciális attribútumain *éppúgy* rendelkezik értékekkel, mint azokon az attribútumain, amelyekkel a szuperosztály tagjaként rendelkezik. Azt mondjuk, hogy egy egyed, amely egy alosztály tagja, **örökli** az egyednek mint a szuperosztály tagjának az összes attribútumát. Az egyed örökli az összes olyan kapcsolatot is, amelyben a szuperosztály részt vesz. Vegyük észre, hogy egy alosztály a saját speciális (vagy lokális)

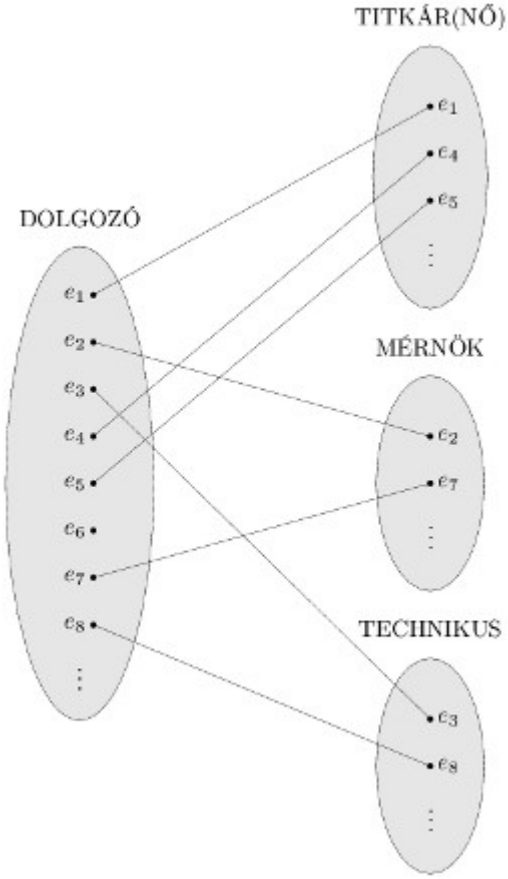
attribútumaival és kapcsolataival, valamint a szuperosztályból örökölt attribútumaival és kapcsolataival egy saját jogú *egyedtípusnak* tekinthető.[19]

Specializáció és generalizáció

Specializáció

A **specializáció** az a folyamat, amelynek során egy egyedtípus *alosztályainak egy halmazát* definiáljuk; ezt az egyedtípust a specializáció **szuperosztályának** nevezzük. A specializációt alkotó alosztályok halmazát a szuperosztályban lévő egyedek valamilyen megkülönböztető jellemzője alapján definiáljuk. A {TITKÁR(NŐ), MÉRNÖK, TECHNIKUS} például a DOLGOZÓ szuperosztály egy olyan specializációja, amely a DOLGOZÓ egyedeket azok *foglalkozása* alapján különbözteti meg. Ugyanannak az egyedtípusnak számos specializációját adhatjuk meg, más-más megkülönböztető tulajdonságuk szerint. A DOLGOZÓ egyedtípus egy másik lehetséges specializációja például a {FIX_FIZETÉSŰ_DOLGOZÓ, ÓRABÉRES_DOLGOZÓ} alosztályhalmazt eredményezi; ez a specializáció a dolgozókat a *fizetési módjuk* alapján különbözteti meg egymástól.

4.2. ábra - Egy specializáció példányai.

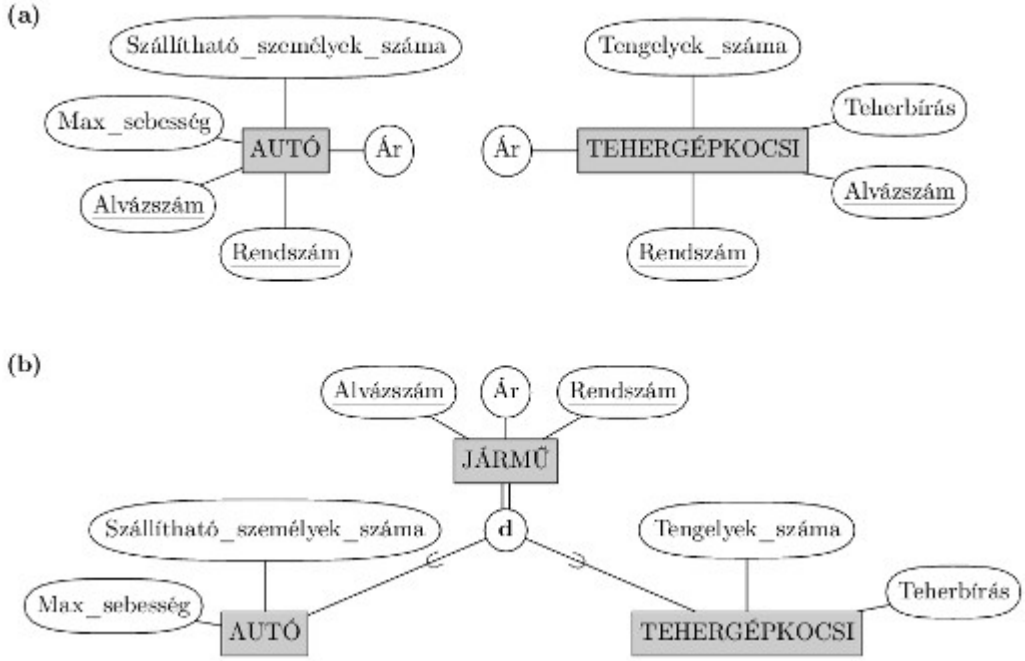


Generalizáció

Elképzelhetjük az absztrakciónak egy *inverz folyamatát*, amelynek során elhagyjuk a különbségeket több egyedtípus között, azonosítjuk a közös jellemzőiket és **generalizáljuk** (általánosítjuk) őket egyetlen **szuperosztályba**, amelynek az eredeti egyedtípusok speciális **alosztályai**. Tekintsük például a 4.3. (a) ábrán látható AUTÓ és TEHERGÉPKOCSI egyedtípusokat! Mivel ezek számos közös attribútummal rendelkeznek, generalizálhatjuk őket a JÁRMŰ egyedtípusba, ahogy a a 4.3.

(b) ábrán látható. Mind az AUTÓ, mind a TEHERGÉPKOCSI a JÁRMŰ **generalizált szuperosztály** alosztályai lesznek. A **generalizáció** kifejezéssel hivatkozunk adott egyedtípusokból egy generalizált egyedtípus kialakításának a folyamatára.

4.3. ábra - Generalizáció. (a) Két egyedtípus, az AUTÓ és a TEHERGÉPKOCSI. (b) Az AUTÓ és a TEHERGÉPKOCSI generalizációja a JÁRMŰ szuperosztályba.



Vegyük észre, hogy a generalizáció folyamatára a specializáció folyamatának funkcionális inverzeként tekinthetünk. Így a 4.3. ábrán az {AUTÓ, TEHERGÉPKOCSI} a JÁRMŰ specializációjának is tekinthető, ahelyett hogy a JÁRMŰ-vet az AUTÓ és a TEHERGÉPKOCSI generalizációjának tekintenénk. Hasonlóan, a 4.1. ábrán a DOLGOZÓ a TITKÁR(NŐ), a TECHNIKUS és a MÉRNÖK generalizációjának is tekinthető. Bizonyos tervezési módszertanokban a generalizációt és a specializációt grafikus jelöléssel is megkülönböztetik egymástól. A gerenalizált szuperosztály felé mutató nyíl generalizációt reprezentál, míg a specializált alosztályok felé mutató nyilak specializációt reprezentálnak. Mi *nem* használjuk ezt a jelölést, mert annak az eldöntése, hogy melyik folyamat felel meg jobban egy konkrét szituációban, gyakran szubjektív.

Az eddigiekben bemutattuk az alosztályok és a szuperosztály/alosztály kapcsolatok fogalmát, valamint a specializáció és a generalizáció folyamatát. Általánosságban egy szuperosztály vagy egy alosztály azonos típusú egyedek egy kollekcióját reprezentálja, ezért egy *egyedtípust* is leír; emiatt — az egyedtípusokhoz hasonlóan — a szuperosztályokat és az alosztályokat is téglalapokkal jelöljük az EER diagramokon. A következőkben a specializációk és a generalizációk tulajdonságait tárgyaljuk részletesebben.

A specializáció- és generalizáció-hierarchiák megszorításai és jellemzői

A specializációkra és generalizációkra vonatkozó megszorítások

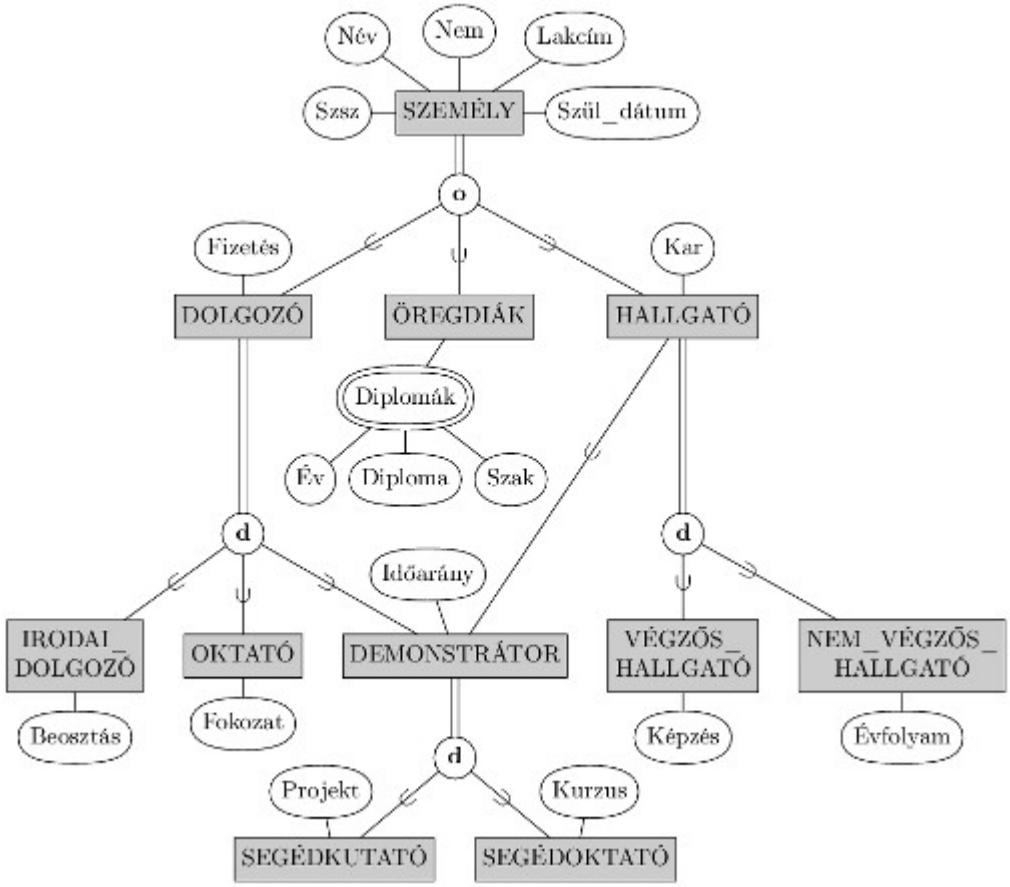
4.4. ábra - EER diagram a Munkakör attribútumdefiniált specializáció jelölésének illusztrálására.

4.5. ábra - EER diagram egy átfedő (nem kizáró) specializáció jelölésének illusztrálására.

Specializációs és generalizációs hierarchiák és hálók

4.6. ábra - Egy specializációs háló a MÉRNÖK_IGAZGATÓ osztott alosztállyal.

4.7. ábra - Az EGYETEM adatbázis specializációs hálója többszörös öröklődéssel.



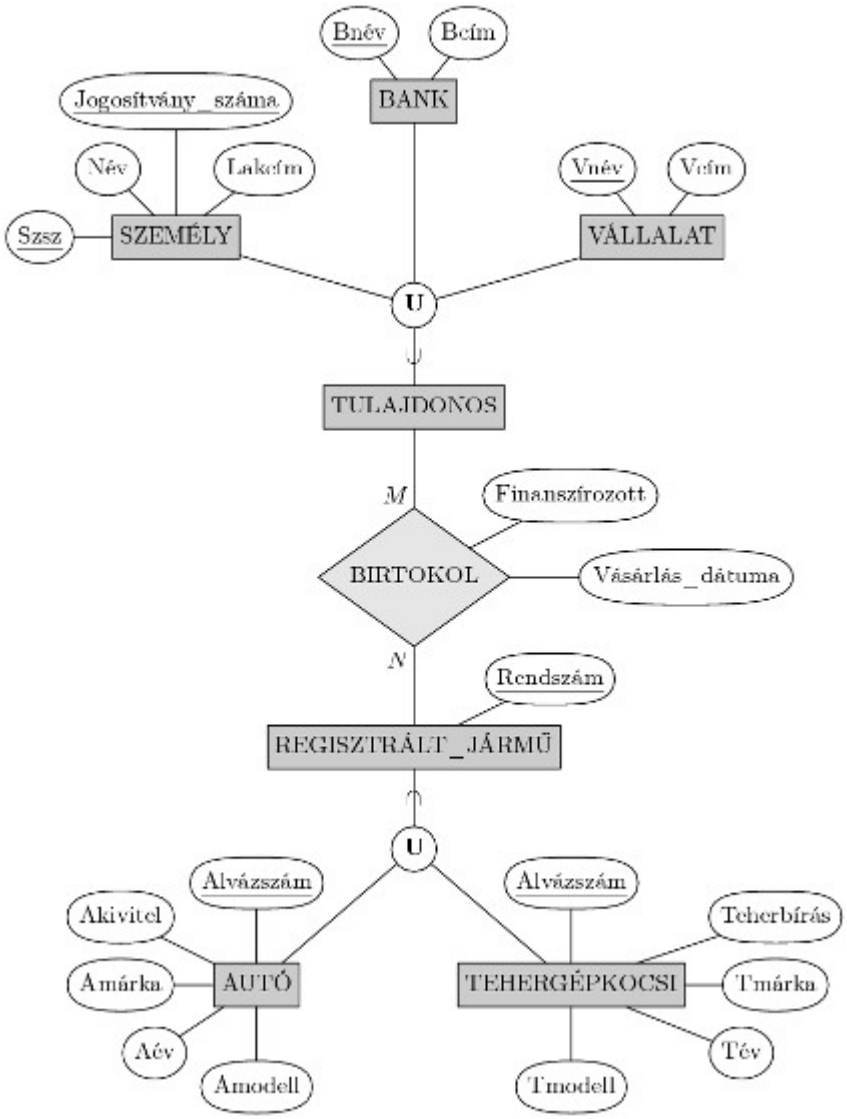
A specializáció és a generalizáció használata a koncepcionális sémák finomításakor

Az unió típusok modellezése kategóriák felhasználásával

Az eddigiekben látott összes szuperosztály/alosztály kapcsolat *egyetlen szuperosztállyal* rendelkezett. Egy osztott alosztály, mint például a 4.6. ábra hálóján látható MÉRNÖK_IGAZGATÓ három *különböző* szuperosztály/alosztály kapcsolatban is alosztály, ahol mindhárom kapcsolatnak

egyetlen szuperosztálya van. Nem ritkán azonban felmerül az igény olyan szuperosztály/alosztály kapcsolat modellezésére, amely *egynél több* szuperosztállyal rendelkezik, ahol a szuperosztályok különböző egyedtípusokat reprezentálnak. Ebben az esetben az alosztály objektumok olyan gyűjteményét reprezentálja, amely a különböző egyedtípusok uniójának a részhalmaza; az ilyen *alosztályt* **unió típusnak** vagy **kategóriának**[20] nevezzük.

4.8. ábra - Két kategória (unió típus): TULAJDONOS és REGISZTRÁLT_JÁRMŰ.



Az EGYETEM adatbázis EER sémája, tervezési lehetőségek és formális definíciók

Ebben az alfejezetben először bemutatunk egy példát egy EER modellbeli adatbázissémára, amely az ebben és a 3. fejezetben tárgyalt különféle fogalmak használatát illusztrálja. Ezután koncepcionális sémák tervezési lehetőségeiről tárgyalunk, majd végül összegezzük az EER modell fogalmait, és definiáljuk őket formálisan ugyanúgy, ahogyan az alap ER modell fogalmait definiáltuk a 3. fejezetben.

Az EGYETEM adatbázis példa

A példa adatbázis alkalmazásunkhoz tekintsük az EGYETEM adatbázist, amely nyilvántartja a

hallgatókat, azok szakjait, teljesítéseit és kurzusfelvételeit, valamint az egyetem által meghirdetett kurzusokat. Az adabázis az oktatók és a végzős hallgatók szponzorált kutatási projektjeit is nyilvántartja. A séma a 4.9. ábrán látható. A következőkben azokat a követelményeket tárgyaljuk, amelyek ezt a sémát eredményezték.

4.9. ábra - EER koncepcionális séma az EGYETEM adatbázishoz.

Az adatbázis minden személyről nyilvántartja a személy nevét (Név), személyi számát (Szsz), címét (Lakcím), nemét (Nem) és születési dátumát (Szdátum). A SZEMÉLY egyedtípusnak két alosztályát különböztetjük meg: OKTATÓ és HALLGATÓ. Az OKTATÓ saját attribútumai a fokozata (Fokozat) (tanársegéd, adjunktus, docens, kutató, vendégoktató stb.), a szobája (Oszoba), az irodai telefonja (Otelefon) és a fizetése (Fizetés). Minden oktató kapcsolatban áll azzal a karral, amelynek dolgozik (DOLGOZIK) (egy oktató több karhoz is tartozhat, a kapcsolattípus ezért M:N számosságú. A HALLGATÓ saját attribútuma az évfolyama (Évfolyam) (1-től 5-ig). Minden hallgató kapcsolatban áll a főszakját és a minorját gondozó karral is, ha azok ismertek (FŐSZAK és MINOR), az aktuális félévben felvett kurzusokkal (FELVETTE), valamint a már teljesített kurzusokkal (TELJESÍTETTE). A TELJESÍTETTE kapcsolattípus minden előfordulása tartalmazza azt az érdemjegyet, amit a hallgató az adott kurzusra kapott (Jegy).

A VÉGZŐS_HALLGATÓ a HALLGATÓ alosztálya, amelyet az Évfolyam = 5 predikátum definiál. Minden végzős hallgató esetén nyilvántartjuk a korábbi diplomáinak a listáját egy összetett többértékű attribútumban (Diplomák). A végzős hallgatók egy témavezető oktatóval (TÉMAVEZETŐ) és egy záróvizsga-bizottsággal (BIZOTTSÁG) is kapcsolatban állnak, ha léteznek ilyenek.

Egy kar név (Knév), telefon (Ktelefon) és iroda (Iroda) attribútumokkal rendelkezik, és kapcsolatban áll azzal az oktatóval, aki a vezetője (VEZETI), valamint azzal az intézménnyel, amelyikhez tartozik (IK). Egy intézménynek név (Inév), iroda (Iiroda) és rektor (Rektor) attribútumai vannak.

Egy tantárgynak tantárgykód (Tkód), tantárgynév (Tnév) és tantárgyleírás (Tleírás) attribútumai vannak. Az egyes tantárgyaknak számos kurzusát hirdetik meg, és minden kurzus esetén tároljuk a kurzuskódot (Kurzuskód), illetve azt a tanévet (Tanév) és félévet (Félév)[21], amelyben a kurzust meghirdették. A kurzuskódok egyértelműen azonosítják az egyes kurzusokat. Az aktuális félévben meghirdetett kurzusok a KURZUS AKTUÁLIS_KURZUS alosztályában vannak, amelyet a „Félév = Aktuális_félév és Tanév = Aktuális_tanév” predikátum definiál. Minden kurzus kapcsolódik ahhoz az előadóhoz, aki tanította vagy tanítja azt (OKTAT), ha az az előadó benne van az adatbázisban.

Az ELŐADÓ_KUTATÓ kategória az OKTATÓ és a VÉGZŐS_HALLGATÓ uniójának részhalmaza, és tartalmazza mindazon oktatókat és végzős hallgatókat, akiket támogatnak a tanításban vagy a kutatásban. Végezetül az ÖSZTÖNDÍJ egyedtípus az egyetem által elnyert kutatási ösztöndíjakat és szerződéseket tartja nyilván. Egy ösztöndíj elnevezés (Név), sorszám (Szám), felajánló szervezet (Szervezet) és kezdő dátum (Kezdő_dátum) attribútumokkal rendelkezik. Az ösztöndíjak egy koordinátorhoz (KOORD) és az általa támogatott összes kutatóhoz kapcsolódnak (TÁMOGAT). A TÁMOGAT minden előfordulása tartalmazza a támogatás kezdő dátumát (Kezdet), lejáratának dátumát (Vég) (ha ismert), valamint a támogatott kutató által a projektre fordított idő százalékát (Idő).

A specializáció/generalizáció tervezési lehetőségei

Nem mindig könnyű megtalálni egy adatbázis alkalmazás legmegfelelőbb koncepcionális tervét. A .

alfejezetben bemutattunk néhány olyan tipikus problémát, amelyekkel az adatbázis-tervező találkozik, amikor az egyedtípusok, kapcsolattípusok és attribútumok közül választ, miközben egy konkrét minivilágbeli szituációt ER sémával reprezentál. Ebben az alfejezetben a specializáció/generalizáció és a kategória (unió típus) EER fogalmakhoz tartozó tervezési irányelveket és lehetőségeket mutatunk be.

Ahogy a . alfejezetben említettük, a koncepcionális adatbázis-tervezést egy iteratívan finomító folyamatnak kell tekinteni, amely addig tart, amíg a számunkra legalkalmasabb tervet el nem érjük. A következő irányelvek segíthetnek az EER fogalmakhoz kötődő tervezési folyamat végrehajtásában:

- Általában sok specializációt tudunk definiálni, hogy a koncepcionális modellünk pontosabb legyen. Ennek hátránya azonban, hogy a terv túl zsúfolt lesz.
- Ha egy alosztálynak csak kevés saját (lokális) attribútuma van, és nincs saját kapcsolata, akkor beleolvaszthatjuk a szuperosztályba. A saját attribútumok NULL értékeket fognak tartalmazni azon egyedekben, amelyek nem tagjai az alosztálynak. Egy *típus* attribútum mondhatja meg, hogy egy egyed tagja-e az alosztálynak.
- Hasonlóan, ha egy specializáció/generalizáció összes alosztályának kevés saját attribútuma van, és nincs saját kapcsolatuk, akkor beleolvaszthatjuk őket a szuperosztályba, és helyettesíthetjük őket egy vagy több *típus* attribútummal, amelyek megadják azt az alosztályt vagy alosztályokat, amely(ek)hez az egyes egyedek tartoznak.
- Az unió típusokat és a kategóriákat általában célszerű elkerülni, hacsak a szituáció ezt a konstrukciótípust nem indokolja határozottan, ami bizonyos gyakorlati helyzetekben előfordul. Ha lehetséges, próbáljunk meg specializáció/generalizáció használatával modellezni, ahogy a . alfejezet végén is írtuk.
- A specializációra/generalizációra vonatkozó kizáró/átfedő, illetve totális/részleges megszorítások választása a modellezett minivilágban létező szabályok alapján történik. Ha a követelmények nem jeleznek semmilyen konkrét megszorítást, akkor az alapértelmezés általában az átfedő és részleges, mivel ez nem ad semmilyen megkötést az alosztályokhoz való tartozásra vonatkozóan.

Ezen irányelvek alkalmazására példaként tekintsük a 4.6. ábrán látható sémát, ahol semmilyen saját (lokális) attribútum nincs. Az összes alosztályt beolvaszthatjuk a DOLGOZÓ egyedtípusba, és a következő attribútumokkal egészíthetjük ki azt:

- Egy Munkakör attribútummal, amelynek a {'Titkár(nő)', 'Mérnök', 'Technikus'} értékhalmaza jelzi, hogy az első specializációban az egyes dolgozók melyik alosztályba tartoznak.
- Egy Fizetési_mód attribútummal, amelynek a {'Fix fizetésű', 'Órabéres'} értékhalmaza jelzi, hogy a második specializációban az egyes dolgozók melyik alosztályba tartoznak.
- Egy Vezető_e attribútummal, amelynek az {'Igen', 'Nem'} értékhalmaza jelzi, hogy egy konkrét alkalmazott egyed vezető-e vagy sem.

Az EER modell fogalmainak formális definíciói

Most összefoglaljuk az EER modell fogalmait, és megadjuk formális definícióikat. Az **osztály**[22] egyedeknek egy halmaza vagy kollekciója; ez magában foglal minden olyan EER sémabeli konstrukciót, amely egyedeket csoportosít, mint például az egyedtípusok, az alosztályok, a szuperosztályok és a kategóriák. Az *S* **alosztály** egy olyan osztály, amelynek az egyedei mindig egy másik osztály részhalmazát alkotják, amely osztályt a **szuperosztály/alosztály** (vagy „**IS-A**”) **kapcsolat** *C* **szuperosztályának** nevezünk. Az ilyen kapcsolatokat *C*/*S*-sel jelöljük. Az ilyen

szuperosztály/alosztály kapcsolatokra mindig teljesül, hogy

$S \subseteq C$.

A $Z = \{S_1, S_2, \ldots, S_n\}$ **specializáció** olyan alosztályok halmaza, amelyek ugyanazzal a G szuperosztállyal rendelkeznek; azaz a G/S_i egy szuperosztály/alosztály kapcsolat minden $i = 1, 2, \ldots, n$ esetén. G-t **generalizált egyedtípusnak** (vagy a specializáció **szuperosztályának**, vagy az $\{S_1, S_2, \ldots, S_n\}$ alosztályok **generalizációjának**) nevezzük.

Z-t **totálisnak** mondjuk, ha mindig (minden pillanatban) teljesül, hogy

$$\bigcup_{i=1}^{n} S_i = G$$

.

Egyébként Z-t **részlegesnek** nevezzük. Z **kizáró**, ha mindig teljesül, hogy

$S_i \cap S_j = \emptyset$ (üres halmaz) bármely $i \neq j$ esetén.

Egyébként Z-t **átfedőnek** nevezzük.

A C S alosztályát **predikátumdefiniáltnak** nevezzük, ha egy, a C attribútumain felírt p predikátum segítségével adjuk meg, hogy C mely egyedei elemei S-nek is; azaz $S = C[p]$, ahol $C[p]$ C azon egyedeinek halmaza, amelyek kielégítik p-t. Azokat az alosztályokat, amelyeket nem predikátummal definiálunk, **felhasználó által definiáltnak** nevezzük.

A Z specializációt (vagy G generalizációt) **attribútumdefiniáltnak** nevezzük, ha egy ($A = c_i$) predikátummal definiáljuk, hogy Z mely egyedei elemei S_i-nek is, ahol A G egy attribútuma, c_i pedig egy konstans érték A tartományából. Vegyük észre, hogy ha $i \neq j$ esetén $c_i \neq c_j$, és A egyértékű attribútum, akkor a specializáció kizáró lesz.

A T **kategória** egy olyan osztály, amely n definiáló szuperosztály ($D_1, D_2, \ldots, D_n$, $n > 1$) uniójának részhalmaza, és formálisan a következőképpen definiáljuk:

$T \subseteq (D_1 \cup D_2 \cup \ldots \cup D_n)$.

A D_i attribútumaira felírt p_i predikátum használható az egyes D_i-k azon elemeinek megadására, amelyek T-nek is elemei. Ha minden D_i-re megadunk egy ilyen predikátumot, akkor a következőt kapjuk:

$T = (D_1[p_1] \cup D_2[p_2] \cup \ldots \cup D_n[p_n])$.

Ezek után kibővítjük a **kapcsolattípus** 3. fejezetben megadott definícióját azzal, hogy megengedjük, hogy bármilyen osztály — nemcsak bármilyen egyedtípus — részt vehessen a kapcsolatban. Ki kell tehát cserélnünk abban a definícióban az *egyedtípus* szó minden előfordulását az *osztály* szóra. Az EER grafikus jelölése konzisztens az ER-ével, mert minden osztályt téglalappal jelölünk.

Összefoglalás

Áttekintő kérdések

1. Mi az az alosztály? Mikor van szükség az adatmodellezésben alosztályra?
2. Definiálja a következő fogalmakat: *egy alosztály szuperosztálya*, *szueprosztály/alosztály kapcsolat*, *IS-A kapcsolat*, *specializáció*, *generalizáció*, *kategória*, *saját (lokális) attribútum* és *saját kapcsolat*!

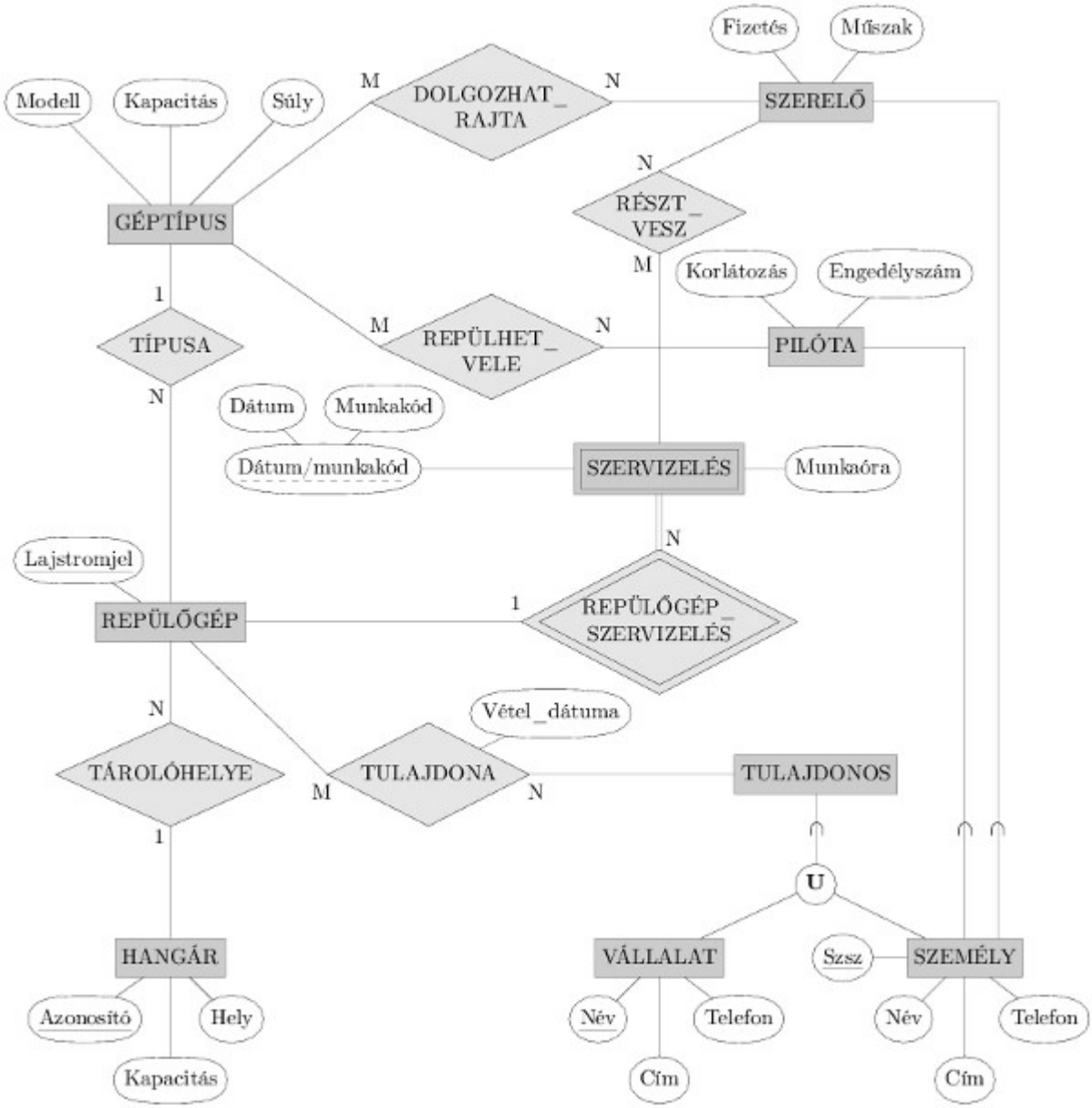
3. Ismertesse az attribútumok/kapcsolatok öröklődési mechanizmusát! Miért hasznos ez?
4. Ismertesse a felhasználó által definiált és a predikátumdefiniált alosztályokat, valamint a közöttük lévő különbségeket!
5. Ismertesse a felhasználó által definiált és a predikátumdefiniált specializációkat, valamint a közöttük lévő különbségeket!
6. Ismertesse a specializációkra és generalizációkra vonatkozó megszorítások két fő típusát!
7. Mi a különbség a specializációs hierarchia és a specializációs háló között?
8. Mi a különbség a specializáció és a generalizáció között? Miért nem jelöljük ezt a különbséget a sémadiagramokon?
9. Miben különbözik a kategória egy hagyományos osztott alosztálytól? Mire használjuk a kategóriákat? Illusztrálja példákkal válaszát!

Feladatok

1. Tervezzen EER sémát egy önt érdeklő adatbázis alkalmazáshoz! Adja meg az összes megszorítást, amelynek teljesülnie kell az adatbázison! A sémában legalább öt egyedtípus, négy kapcsolattípus, egy gyenge egyedtípus, egy szuperosztály/alosztály kapcsolat, egy kategória és egy *n*-edfokú ($n > 2$) kapcsolattípus szerepeljen!
2. A 4.10. ábra egy kis magánrepülőtér adatbázisának az EER diagramját ábrázolja. Az adatbázis nyilvántartja a repülőgépeket, a tulajdonosaikat, a reptéri dolgozókat és a pilótákat. Az adatbázis követelményei közül az alábbi információkat gyűjtöttük össze: Minden REPÜLŐGÉP-nek van egy lajstromjele [Lajstromjel], egy konkrét géptípussal rendelkezik [TÍPUSA], és egy adott hangárban van tárolva [TÁROLÓHELYE]. Minden GÉPTÍPUS-hoz tartozik egy modelljelzés [Modell], egy kapacitás [Kapacitás] és egy súly [Súly]. Minden HANGÁR-nak van egy azonosítószáma [Azonosító], egy kapacitása [Kapacitás] és egy helyszíne [Hely]. Az adatbázis a repülőgépek TULAJDONOS-ait [TULAJDONA], valamint azokat a SZERELŐ-ket is nyilvántartja, akik karbantartják a gépet [RÉSZT_VESZ]. A TULAJDONA kapcsolat minden előfordulása egy REPÜLŐGÉP-et kapcsol össze egy TULAJDONOS-sal, és magában foglalja a vásárlás dátumát is [Vétel_dátuma]. A RÉSZT_VESZ kapcsolat minden előfordulása egy SZERELŐ-t kapcsol össze egy szervizelési munkával [SZERVIZELÉS]. Minden repülőgép számos alkalommal esik át szervizelésen; ezért számos SZERVIZELÉS-sel áll kapcsolatban a [REPÜLŐGÉP_SZERVIZELÉS] kapcsolaton keresztül. Egy SZERVIZELÉS attribútumként tartalmazza a karbantartás dátumát [Dátum], a munkára fordított órák számát [Munkaóra] és az elvégzett munka típusát [Munkakód]. A [SZERVIZELÉS]-t gyenge egyedtípusként vesszük fel a repülőgépek szervizelésének reprezentálására, mert a repülőgép lajstromjelét használjuk a szervizelési munkák azonosítására. A TULAJDONOS vagy egy személy, vagy egy vállalat. Ezért egy unió típust (kategóriát) használunk [TULAJDONOS], amely a vállalat [VÁLLALAT] és a személy [SZEMÉLY] egyedtípusok uniójának a részhalmaza. Mind a pilóták [PILÓTA], mind a szerelők [SZERELŐ] a SZEMÉLY alosztályai. Minden PILÓTA az engedélyszám [Engedélyszám] és a korlátozás [Korlátozás], minden SZERELŐ pedig a fizetés [Fizetés] és a műszak [Műszak] saját attribútumokkal rendelkezik. Az adatbázis minden SZEMÉLY egyedéről tároljuk a személyi számát [Szsz], a nevét [Név], a címét [Cím] és a telefonszámát [Telefon]. A VÁLLALAT egyedekről tárolt információk a név [Név], a cím [Cím] és a telefonszám [Telefon]. Az adatbázis azt is nyilvántartja, hogy az egyes pilótáknak mely repülőgéptípusokra van engedélye [REPÜLHET_VELE], valamint hogy az egyes szerelők mely repülőgéptípusokon végezhetnek karbantartási munkákat [DOLGOZHAT_RAJTA]. Mutassa meg, hogyan lehet a KIS_REPÜLŐTÉR 4.10. ábrán látható EER sémáját UML jelöléssel reprezentálni!

(*Megjegyzés*: nem volt szó arról, hogy hogyan kell a kategóriákat (unió típusokat) UML-ben reprezentálni, ezért nem szükséges a kategóriákat leképezni ebben és a következő feladatban.)

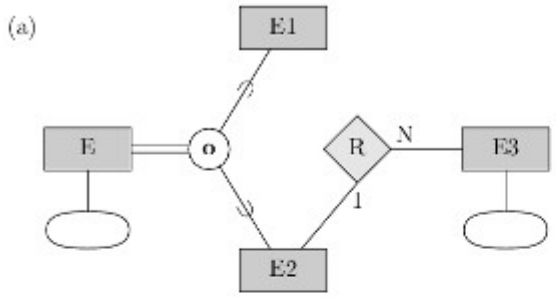
4.10. ábra - EER séma a KIS_REPÜLŐTÉR adatbázishoz.

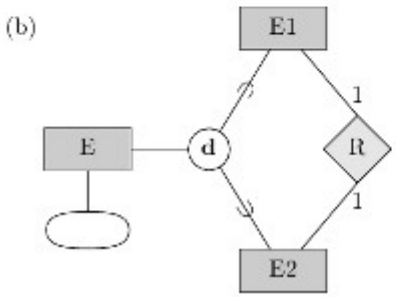


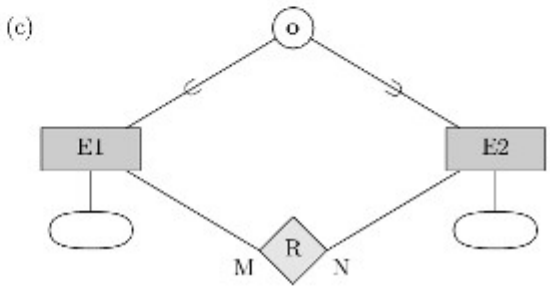
3. Mutassa meg, hogyan lehet az EGYETEM 4.9. ábrán látható EER sémáját UML jelöléssel reprezentálni!
4. Tekintse az alábbi táblázatban látható egyedhalmazokat és attribútumokat! Tegyen minden sor azon oszlopába egy pipát, amely jelzi a bal és a jobb oldali oszlopok közötti megfelelő kapcsolatot!
    a. A bal oldal kapcsolatban áll a jobb oldallal.
    b. A jobb oldal attribútuma a bal oldalnak.
    c. A bal oldal specializációja a jobb oldalnak.
    d. A bal oldal generalizációja a jobb oldalnak.

Egyedhalmaz	(a) kapcsolatban áll	(b) attribútuma	(c) specializációja	(d) generalizációja	Egyedhalmaz vagy attribútum
1. ANYA					SZEMÉLY
2. LEÁNYGYERMEK					ANYA
3. HALLGATÓ					SZEMÉLY
4. HALLGATÓ					Neptun-kód
5. ISKOLA					DIÁK
6. ISKOLA					TANTEREM
7. ÁLLAT					LÓ
8. LÓ					Fajta
9. LÓ					Életkor
10. DOLGOZÓ					Személy_szám
11. BÚTOR					SZÉK
12. SZÉK					Súly
13. EMBER					NŐ
14. KATONA					SZEMÉLY
15. ELLENSÉGES_HARCOS					SZEMÉLY

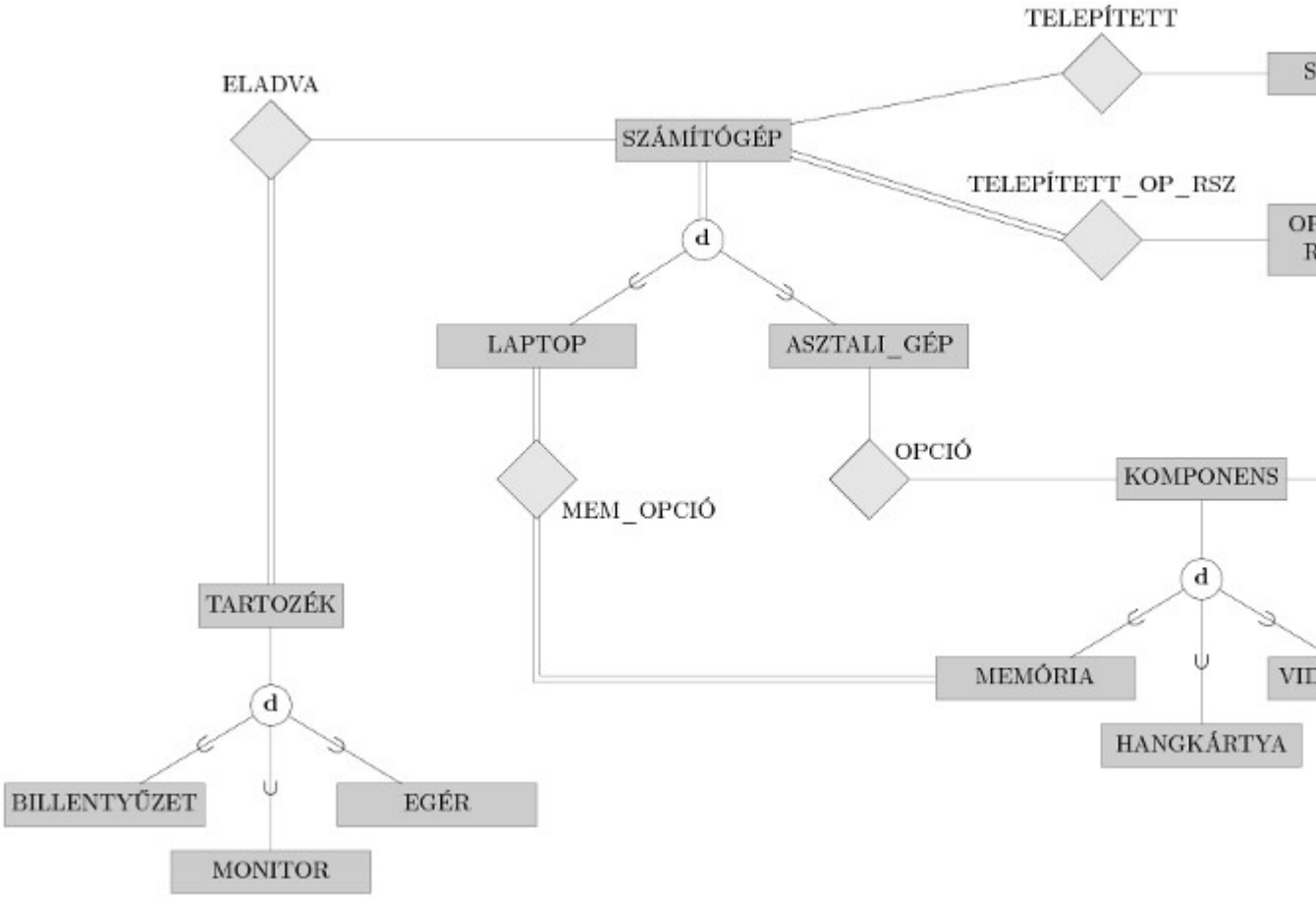
5. Az alábbi EER diagramok közül melyek helytelenek és miért? Írja fel világosan az esetlegesen megfogalmazódó feltételezéseit!







6. Tekintse a következő EER diagramot, amely egy vállalat számítógépes rendszereit írja le! Adjon meg megfelelő attribútumokat és kulcsokat minden egyes egyedtípushoz! Írja fel a max számossági megszorításokat, igazolva választásait! Írjon egy teljes szövegszerű leírást arról, hogy mit reprezentál az így kiegészített EER diagram!



[16] A CAD/CAM a számítógéppel segített tervezés és a számítógéppel segített gyártás kifejezések rövidítése.

[17] Az EER modell neve az *Enhanced* (*kibővített*) ER vagy az *Extended* (*kiterjesztett*) ER elnevezés rövidítéséből származik.

[18] Az osztály/alosztály kapcsolatot gyakran **IS-A** (vagy **IS-AN**) **kapcsolatnak** is nevezzük az alapján, ahogyan a fogalomra utalni szoktunk. Azt mondjuk, hogy a VEZETŐ *az egy* DOLGOZÓ, a TECHNIKUS *az egy* DOLGOZÓ stb.

[19] Egyes objektumorientált programozási nyelvekben általános megszorítás, hogy egy egyed (vagy objektum) *csak egy típussal* rendelkezik. A koncepcionális adatbázis-modellezésben ez általában túl nagy megszorítást jelent.

[20] Az általunk használt *kategória* kifejezés az ECR (egyed-kategória-kapcsolat, Entity-Category-Relationship) modellből származik (Elmasri et al. (1985)).

[21] Feltételezzük, hogy ezen az egyetemen *féléves* rendszert használunk, nem pedig *trimeszteres* vagy *negyedéves* rendszert.

[22] Az *osztály* szó használata itt eltér az olyan objektumorientált programozási nyelvekben használt jelentésétől, mint például a C++. A C++-ban az osztály egy strukturálttípus-definíció a rá alkalmazható függvényekkel (műveletekkel) együtt.

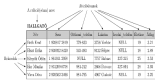
5. fejezet - A relációs adatmodell és a modell megszorításai

Tartalom

A relációs modell alapfogalmai

Tartományok, attribútumok, rekordok és relációk

5.1. ábra - Egy HALLGATÓ reláció attribútumai és rekordjai.



A relációk jellemzői

5.2. ábra - Az 5.1. ábra HALLGATÓ relációja a rekordok más sorrendjében.

Jelölések a relációs modellben

A következő jelöléseket fogjuk használni jegyzetünkben:

- Az $R(A_1, A_2, \ldots, A_n)$ egy n-edfokú R relációsémát jelöl.
- A Q, R, S betűk relációnevet jelölnek.
- A q, r, s betűk relációállapotokat jelölnek.
- A t, u, v betűk rekordokat jelölnek.
- Általában egy relációséma neve, mint például az, hogy HALLGATÓ, az adott relációban szereplő rekordok aktuális halmazát — az *aktuális reláció állapotot* — is jelzi, míg a

HALLGATÓ(Név, Szsz, …) *kizárólag* a relációsémára hivatkozik.

- Egy *A* attribútum a pontozott *R.A*jelölést használva minősíthető azzal az *R* relációnévvel, amelyhez tartozik, például HALLGATÓ.Név vagy HALLGATÓ.Életkor. Ez azért lényeges, mert különböző relációkban szereplő attribútumoknak lehet azonos a nevük. Ugyanakkor *egy adott reláción belül* minden attribútumnévnek különbözőnek kell lennie.
- A $t = \langle v_1, v_2, \ldots v_n \rangle$ egy $r(R)$ relációbeli t elem n-est jelöl, ahol v_i az A_i attribútumnak megfelelő érték. A következő jelölés a rekordok **komponensértékeire** hivatkozik:
  - Mind $t[A_i]$, mind pedig $t.A_i$ (és néha $t[i]$ is) az A_i attribútumnak megfelelő t-beli v_i értékre hivatkozik.
  - Mind $t[A_u, A_w, \ldots, A_z]$, mind pedig $t.(A_u, A_w, \ldots, A_z)$, ahol $A_u, A_w, \ldots, A_z$ R-beli attribútumoknak egy listája, a listában megadott attribútumoknak megfelelő t-beli $\langle v_u, v_w, \ldots, v_z \rangle$ értékű részrekordra hivatkozik.

Példaképpen tekintsük az 5.1. ábra HALLGATÓ relációjában szereplő t = ⟨'Virra Dóra', '2 920502 3496', '881-795', '4967 Csaholc', NULL, 19, 3.25⟩ rekordot; ez alapján t[Név] = ⟨'Virra Dóra'⟩, illetve t[Szsz, Átlag, Életkor] = ⟨'2 860502 3495', 3.25, 19⟩.

A relációs modell megszorításai és a relációs adatbázisséma

Tartománymegszorítások

A tartománymegszorítások előírják, hogy az egyes rekordokon belül minden egyes A attribútum értékének egy atomi értéknek kell lennie a dom(A) tartományból. Az . alfejezetben már tárgyaltuk, hogy milyen módon írhatók le a tartományok. A tartományokhoz társított adattípusok közé tartoznak tipikusan az egész és a valós számok esetén használt standard numerikus adattípusok (ilyenek a short integer, az integer, a long integer, illetve a float és a dupla pontosságú float). Szintén rendelkezésre állnak a karakter, a logikai, a fix és változó hosszúságú sztring típusok, csakúgy, mint a dátum, az idő, az időbélyeg és pénz, valamint egyéb speciális adattípusok. Más lehetséges tartományok leírhatók egy adatttípushoz tartozó értékek résztartományaként vagy egy felsorolásos adattípussal, amelynél az összes lehetséges érték explicit módon meg van adva.

Kulcsmegszorítások és a NULL értékekre vonatkozó megszorítások

Relációs adatbázisok és relációs adatbázissémák

5.3. ábra - Sémadiagram a VÁLLALAT relációs adatbázissémához.

DOLGOZÓ

Vnév	Knév	Szsz	Szdátum	Lakcím	Nem	Fizetés	Főnök_szsz	Osz

OSZTÁLY

Onév	Oszám	Vez_szsz	Vez_kezdő_dátum

OSZT_HELYSZÍNEK

Oszám	Ohelyszín

PROJEKT

Pnév	Pszám	Phelyszín	Osz

DOLGOZIK_RAJTA

Dszsz	Psz	Órák

HOZZÁTARTOZÓ

Dszsz	Hozzátartozó_név	Nem	Szdátum	Kapcsolat

Egyedintegritás, hivatkozási integritás és külső kulcsok

Az **egyedintegritási megszorítás** kimondja, hogy az elsődleges kulcs értéke nem lehet NULL. Ez azért van így, mert az elsődleges kulcs értékét az egyes rekordok azonosítására használjuk egy relációban. Megengedve a NULL értékeket az elsődleges kulcs számára, nem tudnánk egyértelműen azonosítani minden rekordot. Ha például két vagy több rekordnak is NULL lenne az elsődleges kulcsa, akkor nem tudnánk megkülönböztetni őket, amikor megpróbálnánk más relációkból rájuk hivatkozni.

A kulcsmegszorítások és az egyedintegritási megszorítások az egyes relációkra vonatkoznak. A **hivatkozási integritási megszorítást** két reláció között értelmezzük, és a két relációban lévő rekordok közötti konzisztencia megteremtése érdekében használjuk. Informálisan a hivatkozási integritási megszorítás kimondja, hogy egy reláció egy rekordjának, amely egy másik relációra hivatkozik, egy *létező rekordra* kell hivatkoznia abban a relációban. Például az 5.4. ábrán a DOLGOZÓ Osz attribútuma annak az osztálynak a számát adja meg, ahol az egyes dolgozók dolgoznak; ezért az értékének minden DOLGOZÓ rekordban meg kell egyeznie az OSZTÁLY reláció valamely rekordjának Oszám értékével.

A hivatkozási integritási megszorítás formálisabb definíciójának a megadásához először definiáljuk a *külső kulcs* fogalmát. A külső kulcsra az alábbiakban ismertetett feltételek megadnak egy hivatkozási integritási megszorítást két relációséma, R_1 és R_2 között. Az R_1 relációséma egy FK attribútumhalmaza **külső (idegen) kulcsa** R_1-nek, amely hivatkozik az R_2 relációsémára, ha eleget tesz a következő feltételeknek:

1. Az FK-beli attribútumoknak és az R_2 PK-val jelölt kulcs attribútumainak páronként azonos a tartománya; ekkor azt mondjuk, hogy az FK attribútumok **hivatkoznak** az R_2 relációsémára.
2. Bármely $r_1(R_1)$ aktuális állapotának egy t_1 rekordjában egy FK-beli érték *vagy* megjelenik egy $r_2(R_2)$ aktuális állapotának valamely t_2 rekordjában PK értékeként, *vagy az értéke NULL*. Az előbbi esetben $t_1[\text{FK}] = t_2[\text{PK}]$, ekkor azt mondjuk, hogy a t_1 rekord **hivatkozik** a

t_2 rekordra.

Ebben a definícióban R_1-et **hivatkozó**, R_2-t **hivatkozott relációnak** nevezzük. Ha e két feltétel teljesül, egy **hivatkozási integritási megszorítás** áll fenn R_1-ről R_2-re vonatkozóan. Egy sok relációt tartalmazó adatbázisban általában sok hivatkozási integritási megszorítás szokott lenni.

E megszorítások megadásához mindenekelőtt tisztában kell lennünk azzal a jelentéssel vagy szereppel, amit az egyes attribútumhalmazok az adatbázis különféle relációsémáiban játszanak. A hivatkozási integritási megszorítások tipikusan a relációsémákkal reprezentált *egyedek közötti kapcsolatokból* származnak. Tekintsük például az 5.4. ábtán látható adatbázist. A DOLGOZÓ relációban az Osz attribútum arra az osztályra hivatkozik, amelyen a dolgozó dolgozik; ennélfogva az Osz-t a DOLGOZÓ OSZTÁLY relációra hivatkozó külső kulcsának jelöljük. Ez azt jelenti, hogy az Osz értékének a DOLGOZÓ reláció bármely t_1 rekordjában meg kell egyeznie az OSZTÁLY elsődleges kulcsának — az Oszám attribútumnak — az OSZTÁLY reláció valamely t_2 rekordjában szereplő értékével, vagy az Osz értéke *lehet NULL*, ha a dolgozó nem tartozik egyik osztályhoz sem, vagy csak később lesz egy osztályhoz hozzárendelve. Az 5.4. ábrán 'Kovács László' dolgozó rekordja a 'Kutatás' osztály rekordjára hivatkozik, jelezve azt, hogy 'Kovács László' ezen az osztályon dolgozik.

5.4. ábra - Egy, a VÁLLALAT adatbázissémához tartozó lehetséges adatbázis-állapot

DOLGOZÓ

Vnév	Knév	Szsz	Szdátum	Lakcím	Nem	Fizetés	Főnök_szsz	Osz
Kovács	László	1 650109 0812	1965. január 9.	4033 Debrecen	F	390000	2 551208 2219	5
Szabó	Mária	2 551208 2219	1955. december 8.	1097 Budapest	N	520000	1 371110 4519	5
Kiss	István	1 680119 6749	1968. január 19.	1172 Budapest	F	325000	1 410620 4902	4
Takács	József	1 410620 4902	1941. június 20.	4027 Debrecen	F	559000	1 371110 4519	4
Horváth	Erzsébet	2 620915 3134	1962. szeptember 15.	1092 Budapest	N	494000	2 551208 2219	5
Tóth	János	1 720731 2985	1972. július 31.	6726 Szeged	F	325000	2 551208 2219	5
Fazekas	Ilona	2 690329 1099	1969. március 29.	3535 Miskolc	N	325000	1 410620 4902	4
Nagy	Zoltán	1 371110 4519	1937. november 10.	1061 Budapest	F	715000	NULL	1

OSZTÁLY

Onév	Oszám	Vez_szsz	Vez_kezdő_dátum
Kutatás	5	2 551208 2219	1988. május 22.
Humán erőforrás	4	2 690329 1099	1995. január 1.
Központ	1	1 371110 4519	1981. június 19.

OSZT_HELYSZÍNEK

Oszám	Ohelyszín
1	Budapest
4	Kecskemét
5	Vác
5	Tiszafüred
5	Budapest

DOLGOZIK_RAJTA

Dszsz	Psz	Órák
1 650109 0812	1	32.5
1 650109 0812	2	7.5
2 620915 3134	3	40.0
1 720731 2985	1	20.0
1 720731 2985	2	20.0
2 551208 2219	2	10.0
2 551208 2219	3	10.0
2 551208 2219	10	10.0
2 551208 2219	20	10.0
1 680119 6749	30	30.0
1 680119 6749	10	10.0
2 690329 1099	10	35.0
2 690329 1099	30	5.0
1 410620 4902	30	20.0
1 410620 4902	20	15.0
1 371110 4519	20	NULL

PROJEKT

Pnév	Pszám	Phelyszín	Osz
X termék	1	Vác	5
Y termék	2	Tiszafüred	5
Z termék	3	Budapest	5
Komputerizáció	10	Kecskemét	4
Reorganizáció	20	Budapest	1
Új fejlesztések	30	Kecskemét	4

HOZZÁTARTOZÓ

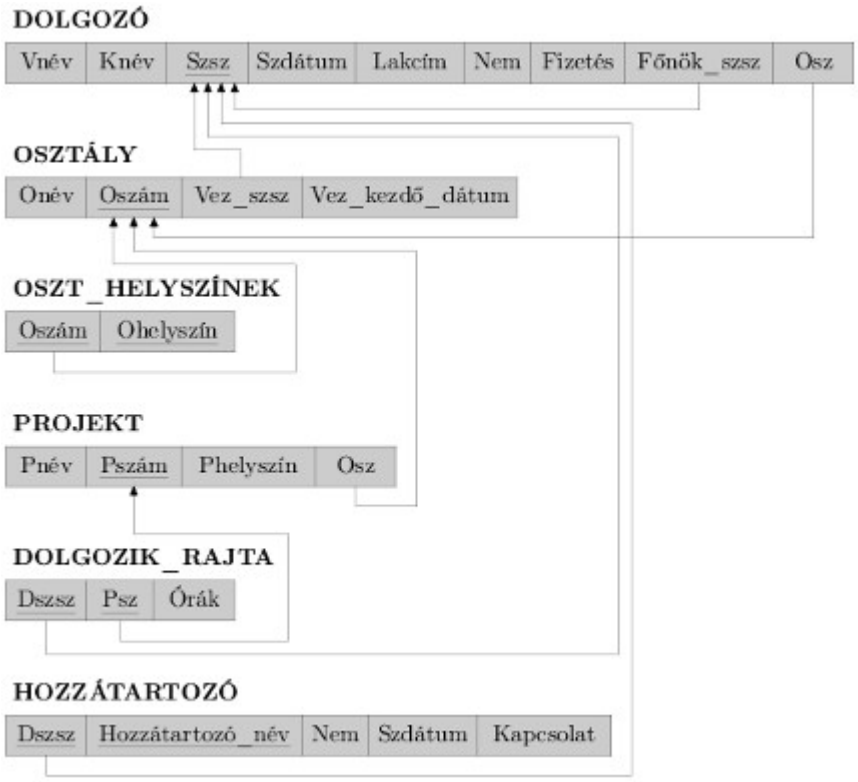
Dszsz	Hozzátartozó_név	Nem	Szdátum	Kapcsolat
2 551208 2219	Anna	N	1986. április 5.	lánya
2 551208 2219	Bence	F	1983. október 25.	fia
2 551208 2219	Máté	F	1958. május 3.	házastársa
1 410620 4902	Viktória	N	1942. február 28.	házastársa
1 650109 0812	Balázs	F	1988. január 4.	fia
1 650109 0812	Anna	N	1988. december 30.	lánya
1 650109 0812	Réka	N	1967. május 5.	házastársa

Megjegyzendő, hogy egy külső kulcs *hivatkozhat a saját relációjára* is. Például a Főnök_szsz attribútum a DOLGOZÓ-ban a dolgozó főnökére hivatkozik; ez egy másik dolgozó, akit a DOLGOZÓ relációban egy rekord szemléltet. Ennélfogva a Főnök_szsz egy olyan külső kulcs, amely a DOLGOZÓ relációra önmagára hivatkozik. Az 5.4. ábrán a 'Kovács László' dolgozóhoz tartózó rekord 'Szabó Mária' dolgozó rekordjára hivatkozik, jelezve, hogy 'Szabó Mária' 'Kovács László' főnöke.

Egy folytonos vonalú nyilat húzva minden egyes külső kulcstól ahhoz a relációhoz, amelyre az adott külső kulcs hivatkozik, *diagramszerűen jeleníthetjük meg az integritási megszorításokat*. A világosság kedvéért, a nyílhegy a hivatkozott reláció elsődleges kulcsára mutathat. Az 5.5. ábra mutatja az 5.3. ábra sémáját az ilyen módon ábrázolt hivatkozási integritási megszorításokkal.

5.5. ábra - Hivatkozási integritási megszorítások megjelenítése a VÁLLALAT relációs

adatbázissémán.



Minden integritási megszorítást meg kell adnunk a relációs adatbázissémára (azaz a definíciója részeként definiálva), ha azt szeretnénk, hogy ezek a megszorítások érvényre legyenek juttatva az adatbázisállapotokban. Ezért a DDL tartalmaz olyan szolgáltatásokat a különféle típusú megszorítások megadásához, hogy a DBMS automatikusan kikényszeríthesse azokat. A legtöbb relációs DBMS támogatja a kulcs- és egyedintegritási megszorításokat, és rendelkezik eszközökkel a hivatkozási integritás támogatására. Ezeket a megszorításokat az adatdefiníció részeként adhatjuk meg.

Egyéb megszorításfajták

Módosítási műveletek, tranzakciók, a megszorítások megsértésének kezelése

A beszúrási művelet

A törlési művelet

A módosítási művelet

A tranzakció fogalma

Összefoglalás

Áttekintő kérdések

1. Definiálja a következő fogalmakat: *tartomány*, *attribútum*, *elem n-es*, *relációséma*, *relációállapot*, *egy reláció foka*, *relációs adatbázisséma* és *relációs adatbázis-állapot*!
2. Miért nem rendezettek a rekordok egy relációban?
3. Miért nem megengedettek a duplikált rekordok egy relációban?
4. Mi a különbség egy kulcs és egy szuperkulcs között?
5. Miért kell kijelölnünk egy reláció kulcsjelöltjeinek egyikét elsődleges kulcsnak?
6. Ismertesse a relációk azon jellemzőit, amelyek megkülönböztetik őket a hétköznapi táblázatoktól és állományoktól!
7. Soroljon fel különféle okokat, amelyek NULL értékek megjelenéséhez vezetnek a relációkban!
8. Ismertesse az egyedintegritási és a hivatkozási integritási megszorítást! Miért tartjuk ezeket fontosnak?
9. Definiálja a *külső kulcsot*! Mire használjuk ezt a fogalmat?
10. Mi az a tranzakció? Miben különbözik egy módosítási művelettől?

Feladatok

1. Tegyük fel, hogy az alábbi módosítási műveleteket közvetlenül alkalmazzuk az 5.4. ábrán látható adatbázis állapotra. Adja meg az *összes* integritási megszorítást, amelyet az egyes műveletek megsértenek, ha vannak ilyenek, és mutassa be ezen megszorítások kikényszerítésének a különböző módjait!
    a. Szúrja be a ⟨'Szöllősi', 'Róbert', '1 741206 3240', '1974. december 6.', '4031 Debrecen', 'F', 754000, '1 371110 4519', 1⟩ rekordot a DOLGOZÓ relációba!
    b. Szúrja be a ⟨'A termék', 4, 'Vác', 2⟩ rekordot a PROJEKT relációba!
    c. Szúrja be a ⟨'Gyártás', 4, '1 741206 3240', '1998. október 1.'⟩ rekordot az OSZTÁLY

relációba!

d. Szúrja be a ⟨'2 820617 6139', NULL, 40.0⟩ rekordot a DOLGOZIK_RAJTA relációba!

e. Szúrja be a ⟨'1 720731 2985', 'Rita', 'N', '1973. december 12.', 'házastárs'⟩ rekordot a HOZZÁTARTOZÓ relációba!

f. Törölje azokat a rekordokat a DOLGOZIK_RAJTA relációból, amelyekben Dszsz = '2 551208 2219'!

g. Törölje azokat a rekordokat a DOLGOZÓ relációból, amelyekben Szsz = '1 410620 4902'!

h. Törölje azokat a rekordokat a PROJEKT relációból, amelyekben Pnév = 'X termék'!

i. Módosítsa az OSZTÁLY reláció Vez_szsz és Vez_kezdő_dátum attribútumait rendre '1 650109 0812'-re és '1990. október 1.'-re abban a rekordban, amelyben Oszám = 5!

j. Módosítsa a DOLGOZÓ reláció Főnök_szsz attribútumát '1 741206 3240'-re abban a rekordban, amelyben Szsz = '1 680119 6749'!

k. Módosítsa a DOLGOZIK_RAJTA reláció Órák attribútumát 5.0-ra abban a rekordban, amelyben Dszsz = '1 680119 6749' és Psz = 10!

2. Tekintse az 5.6. ábrán látható REPÜLŐGÉP_MENETREND relációs adatbázissémát, amely légitársaságok járatinformációinak egy adatbázisát írja le!

5.6. ábra - A REPÜLŐGÉP_MENETREND relációs adatbázisséma.

Minden JÁRAT egy Járatszámmal van azonosítva, és egy vagy több, 1, 2, 3 stb. értékű Szakasz_sorszámmal számozott SZAKASZ-ból áll. Minden szakasz rendelkezik egy menetrend szerinti érkezési és indulási idővel, reptérrel és egy vagy több SZAKASZ_PÉLDÁNY-nyal; eggyel-eggyel minden olyan Dátumra, amelyeken a járat közlekedik. Minden JÁRAT-nál nyilvántartjuk a VITELDÍJ-akat. Minden SZAKASZ_PÉLDÁNY-nál nyilvántartjuk az ÜLŐHELY_FOGLALÁS-okat, mint ahogy azt a REPÜLŐGÉP-et is, amelyet az adott szakaszon használnak, valamint a tényleges érkezési és indulási időket és reptereket. Minden REPÜLŐGÉP-et egy Lajstromjel azonosít, és mindegyiknek egy konkrét GÉPTÍPUS-a van. A LESZÁLLHAT az egyes GÉPTÍPUS-okat köti össze azokkal a REPÜLŐTÉR egyedekkel, amelyeken az adott géptípus képes leszállni. A REPÜLŐTÉR egyedeket egy Reptérkód azonosítja. Tekintsen egy olyan módosítást a REPÜLŐGÉP_MENETREND adatbázison, amely egy adott járatra vagy szakaszra visz fel egy új foglalást egy adott dátummal!

   a. Adja meg ennek a módosításnak a műveleteit!

   b. Mit gondol, milyen típusú megszorításokat kellene ellenőrizni?

   c. Ezen megszorítások közül melyek kulcs-, egyedintegritási és hivatkozási integritási megszorítások, és melyek nem azok?

   d. Adja meg az összes hivatkozási integritási megszorítást, amely az 5.6. ábra sémáján fennáll!

3. Tekintse a TANÓRA(Tárgykód, Kurzuskód, Oktató, Félév, Épület, Terem, Óraszám, Napok, Kredit) relációt! Ez a reláció egyetemi tanórákat reprezentál, amelyek mindegyike egyedi Kurzuskóddal rendelkezik. Mit gondol, mik lehetnek a kulcsjelöltek? Írja le saját szavaival azokat a megszorításokat, amelyek fennállása esetén az egyes kulcsjelöltek érvényesek lehetnek!

6. fejezet - A relációalgebra és a relációkalkulusok

Tartalom

Ebben a fejezetben a relációs modell két formális nyelvét tárgyaljuk: a relációalgebrát és a relációkalkulust. Ahogy a 2. fejezetben tárgyaltuk, az adatmodelleknek — az adatbázis szerkezetének és megszorításainak definiálására szolgáló eszközökön kívül — tartalmazniuk kell olyan műveleteket, amelyekkel az adatbázis manipulálható. A relációs modellre vonatkozó alapműveletek halmaza a **relációalgebra**. Ezek a műveletek lehetővé teszik, hogy a felhasználók alapvető lekérdezéseket adhassanak meg. A lekérdezések eredménye egy új reláció, amelyet egy vagy több relációból kapunk. Az algebra műveletei tehát új relációkat állítanak elő, amelyek tovább manipulálhatók ugyanezen algebra műveleteivel. A relációalgebrai műveletek egy sorozata **relációalgebrai kifejezést** alkot, amelynek az eredménye szintén egy reláció, ami egy adatbázis-lekérdezés eredményét reprezentálja.

A relációalgebra számos okból kifolyólag nagyon fontos. Először is a relációs modell műveleteinek egy formális megalapozását biztosítja. A második, és talán a legfontosabb indok, hogy a lekérdezések implementálásának és optimalizálásának az alapját képezi a relációs adatbázis-kezelő rendszerekben (RDBMS-ekben). Harmadrészt bizonyos elemei az RDBMS-ek SQL standard lekérdező nyelvébe is beépülnek. Bár egyetlen ma kereskedelmi forgalomban lévő RDBMS sem nyújt felületet relációalgebrai lekérdezések készítéséhez, bármely relációs rendszer alapvető műveletei és függvényei a relációalgebrai műveleteken alapulnak. Ezeket a műveleteket részletesen tárgyaljuk a következő alfejezetekben.

Míg az algebra egy művelethalmazt definiál a relációs modellhez, a **relációkalkulus** egy magasabb szintű deklaratív jelölésrendszert biztosít a relációs lekérdezések megadásához. Egy relációkalkulusbeli kifejezés egy új relációt eredményez, amelyet olyan változókkal adunk meg, amik az adatbázisban tárolt relációk sorait (a rekord alapú kalkulusban) vagy oszlopait (a tartomány alapú kalkulusban) járják be. Egy kalkulusbeli kifejezésben nincs megadva egy műveletsorozat, amely megadná, hogy hogyan kell lekérni az eredményt — a kalkulusbeli kifejezés csak azt specifikálja, hogy az eredménynek milyen információkat kell tartalmaznia. Ez a fő különbség a relációalgebra és a relációkalkulus között. A relációkalkulus azért fontos, mert szilárd matematikai logikai alapokkal bír, és mert az RDBMS-ek standard lekérdező nyelve (az SQL) bizonyos elemeket a rekord alapú relációkalkulusból vesz át.[23]

A relációalgebrát gyakran tekintjük a relációs adatmodell szerves részének. A műveletei két csoportra oszthatók. Az egyik csoport a matematikai halmazelmélet halmazműveleteiből áll; ezek azért alkalmazhatók, mert a formális relációs modellben a relációt rekordok halmazaként definiáljuk. A halmazműveletek közé tartozik az unió, a metszet, a (halmaz)különbség és a Descartes-szorzat. A másik csoport olyan műveletekből áll, amelyeket speciálisan a relációs modellhez fejlesztettek ki — ilyen többek között a szelekció, a projekció és az összekapcsolás. Először a szelekció és a projekció műveleteit tárgyaljuk a . alfejezetben, mivel ezek **unáris műveletek**, amelyek egy reláción operálnak. Ezután a halmazműveleteket vesszük a . alfejezetben. A . alfejezetben pedig az összekapcsolással és egyéb összetett **bináris műveletekkel** foglalkozunk, amelyek két táblán operálnak. A példáinkban az 5.4. ábrán látható VÁLLALAT adatbázist használjuk.

Vannak olyan általános adatbázis-lekérdezések, amelyek nem hajthatók végre az eredeti

relációalgebra műveleteivel, így további műveleteket hoztak létre, hogy az ilyen kéréseket is ki tudjuk fejezni. Ilyenek a **csoportosító függvények**, amely műveletekkel a táblák adatait lehet *összegezni*, illetve ilyenek az összekapcsolás és az unió műveletek további típusai is. Ezeket a műveleteket — amelyekkel a . alfejezetben foglalkozunk majd — az adatbázis alkalmazásokban játszott fontos szerepük miatt adták hozzá az eredeti relációalgebrához. Relációs műveleteket használó lekérdezésekre adunk példákat a . alfejezetben. Közülük néhány lekérdezés visszaköszön a későbbi alfejezetekben a különböző nyelvek illusztrálásakor.

A . és . alfejezetekben a relációs adatbázisok másik fő formális nyelvét, a **relációkalkulust** tárgyaljuk. A relációkalkulusnak két változata létezik. A rekord alapú relációkalkulust a ., míg a tartomány alapú relációkalkulust a . alfejezetben tárgyaljuk. A relációkalkulus egy formális nyelv, amely a matematikai logika egyik ágán, a predikátumkalkuluson alapul. A rekord alapú relációkalkulusban a változók a rekordokon, míg a tartomány alapú relációkalkulusban az attribútumok tartományain (értékein) futnak végig. A . alfejezet összefoglalja a fejezetben leírtakat.

Unáris relációs operátorok: a szelekció és a projekció

A szelekció művelete

A szelekció használatával egy relációból kiválaszthatjuk a rekordoknak egy olyan *részhalmazát*, amely eleget tesz egy **szelekciós feltételnek**. Tekinthetünk a szelekcióra úgy is, mint egy *szűrésre*, amely csak azokat a rekordokat tartja meg, amelyek kielégítenek egy minősítő feltételt. A szelekció műveletét úgy is el lehet képzelni, mint a reláció egy *horizontális felosztását* rekordok két halmazára — azokra a rekordokra, amelyek kielégítik a feltételt és ki lesznek választva, illetve azokra a rekordokra, amelyek nem elégítik ki a feltételt, és így nem lesznek kiválasztva. Például azoknak a DOLGOZÓ rekordoknak a kiválasztásánál, amelyeknek az osztálya a 4-es osztály, vagy azoknál, amelyeknél a fizetés nagyobb 390000 forintnál, ezen két feltétel mindegyikét megadhatjuk egy szelekció művelettel a következőképpen:

- $\sigma_{Osz=4}(\text{DOLGOZÓ})$
- $\sigma_{Fizetés>390000}(\text{DOLGOZÓ})$

Általánosságban a szelekció művelete

- $\sigma_{\text{szelekciós feltétel}}(R)$

alakú, ahol a σ (a görög szigma) betűt használjuk a szelekció műveletének a jelölésére, míg a szelekciós feltétel egy logikai kifejezés, amelyet az *R* reláció attribútumaira írunk elő. Megjegyzendő, hogy *R* általában egy *relációalgebrai kifejezés*, amelynek az eredménye egy reláció — a legegyszerűbb ilyen kifejezés éppen egy adatbázisrelációnak a neve. Annak a relációnak, amit a szelekció művelet eredményez, *ugyanazok lesznek az attribútumai*, mint *R*-nek.

Egy, a ⟨szelekciós feltétel⟩-ben megadott logikai kifejezés

- ⟨attribútumnév⟩ ⟨hasonlító operátor⟩ ⟨konstans érték⟩

vagy

- ⟨attribútumnév⟩ ⟨hasonlító operátor⟩ ⟨attribútumnév⟩

alakú **klózokból** épül fel, ahol az ⟨attribútumnév⟩ *R* egy attribútumának a neve, a ⟨hasonlító operátor⟩ rendszerint az {=, <, ≤, >, ≥, ≠} operátorok valamelyike, a ⟨konstans érték⟩ pedig egy konstans érték az attribútum tartományából. A klózok korlátlanul összekapcsolhatók az *and*, *or* és *not* logikai operátorokkal, kialakítva egy általános szelekciós feltételt. Például azon dolgozók rekordjainak a kiválasztásához, akik vagy a 4-es osztályon dolgoznak és 325000 forintnál többet keresnek, vagy pedig az 5-ös osztályon dolgoznak és 390000 forintnál keresnek többet, a következő

szelekció műveletet adhatjuk meg:

- $\sigma_{(\text{Osz}=4 \textbf{ AND } \text{Fizetés}>325000) \textbf{ OR } (\text{Osz}=5 \textbf{ AND } \text{Fizetés}>390000)}(\text{DOLGOZÓ})$

Az eredmény a 6.1. (a) ábrán látható.

6.1. ábra - Szelekció és projekció műveletek eredményei. (a) $\sigma_{(\text{Osz}=4 \text{ AND } \text{Fizetés}>325000) \text{ OR } (\text{Osz}=5 \text{ AND } \text{Fizetés}>390000)}(\text{DOLGOZÓ})$. (b) $\pi_{\text{Vnév, Knév, Fizetés}}(\text{DOLGOZÓ})$. (c) $\pi_{\text{Nem, Fizetés}}(\text{DOLGOZÓ})$.

(a)

Vnév	Knév	Szsz	Szdátum	Lakcím	Nem	Fizetés	Főnök_szsz	Osz
Szabó	Mária	2 551208 2219	1955. december 8.	1097 Budapest	N	520000	1 371110 4519	5
Takács	József	1 410620 4902	1941. június 20.	4027 Debrecen	F	559000	1 371110 4519	4
Horváth	Erzsébet	2 620915 3134	1962. szeptember 15.	1092 Budapest	N	494000	2 551208 2219	5

(b)

Vnév	Knév	Fizetés
Kovács	László	390000
Szabó	Mária	520000
Kiss	István	325000
Takács	József	559000
Horváth	Erzsébet	494000
Tóth	János	325000
Fazekas	Ilona	325000
Nagy	Zoltán	715000

(c)

Nem	Fizetés
F	390000
N	520000
F	325000
F	559000
N	494000
N	325000
F	715000

Megjegyzendő, hogy a {=, <, ≤, >, ≥, ≠} halmazbeli hasonlító operátorok olyan attribútumokra alkalmazhatók, amelyeknek a tartományai *rendezett értékek*, mint például az egészek vagy a dátumok tartományai. A karakterláncok tartományait rendezettnek tekintjük a karakterek ábécé szerinti sorrendjére alapozva. Ha egy attribútum tartománya *rendezetlen értékek* halmaza, akkor csak a {=, ≠} halmazbeli hasonlító operátorok használhatók. Egy rendezetlen tartományra példa a Szín = {'piros', 'kék', 'zöld', 'fehér', 'sárga', …} tartomány, ahol nincs sorrend definiálva a különféle színek között. Egyes tartományok további hasonlító operátorokat is megengednek; például karakterláncok egy tartományán értelmezhető a RÉSZSZTRINGJE hasonlító operátor.

Egy szelekció művelet eredményét általában a következőképpen határozhatjuk meg. A ⟨szelekciós feltétel⟩-t egymástól függetlenül alkalmazzuk minden egyes *R*-beli *t* rekordra. Ezt úgy tesszük meg, hogy az A_i attribútum minden egyes szelekciós feltételbeli előfordulását helyettesítjük az ő $t[A_i]$ rekordbeli értékével. Ha így a feltétel igazzá válik, akkor a *t* rekord **ki lesz választva**. Minden kiválasztott rekord megjelenik a szelekció művelet eredményében. Az AND, OR és NOT logikai műveletek interpretációja a szokásos:

- (felt1 **AND** felt2) akkor igaz, ha mind a (felt1), mind a (felt2) igaz; különben hamis.
- (felt1 **OR** felt2) akkor igaz, ha vagy a (felt1), vagy a (felt2), vagy mindkettő igaz; különben hamis.
- (**NOT** felt) akkor igaz, ha a (felt) hamis; különben hamis.

A szelekció művelete **unáris** művelet, ami azt jelenti, hogy csak egy operandusa van neki (ami egy reláció). A szelekció műveletet ráadásul *minden egyes rekordra külön-külön* alkalmazzuk; így a szelekciós feltétel nem vonatkozhat egyszerre egynél több rekordra. A szelekció műveletének az

eredményeképpen kapott relációnak a **foka** — az attribútumainak a száma — ugyanannyi, mint az *R* foka. Az eredmény reláció rekordjainak a száma mindig *kisebb vagy egyenlő*, mint az *R* rekordjainak a száma. Azaz $|\sigma_C(R)| \leq |R|$ bármilyen *C* feltételre. Egy adott szelekciós feltétellel kiválasztott rekordok arányát a feltétel **szelektivitásának** nevezzük.

Megjegyzendő, hogy a szelekció művelete **kommutatív**, azaz

- $\sigma_{\langle \text{feltétel 1} \rangle}(\sigma_{\langle \text{feltétel 2} \rangle}(R)) = \sigma_{\langle \text{feltétel 2} \rangle}(\sigma_{\langle \text{feltétel 1} \rangle}(R))$.

Ennélfogva szelekciós műveletek egy sorozata tetszőleges sorrendben alkalmazható. Továbbá mindig összekapcsolhatjuk szelekciós műveletek egy **kaszkádját** egyetlen szelekciós műveletté a konjunkciós operátorral (AND), azaz

- $\sigma_{\langle \text{feltétel 1} \rangle}(\sigma_{\langle \text{feltétel 2} \rangle}(\ldots(\sigma_{\langle \text{feltétel } n \rangle}(R))\ldots)) = \sigma_{\langle \text{feltétel 1} \rangle}$ **AND** $\langle \text{feltétel 2} \rangle$ **AND** … **AND** $\langle \text{feltétel } n \rangle (R)$.

A projekció művelete

Ha egy relációra egy táblaként gondolunk, a szelekció művelete néhány *sort* választ ki a táblából, míg a több sort elveti. A **projekció** művelete ugyanakkor bizonyos *oszlopokat* választ ki a táblából, míg a többi oszlopot elveti. Ha csak bizonyos attribútumai érdekelnek minket a relációnak, a projekció műveletét használjuk arra, hogy a relációt *projektáljuk* (levetítsük) ezekre az attribútumokra. Ezért a projekció műveletének eredményét úgy képzelhetjük el, mint a reláció egy *vertikális felosztását* két relációra: az egyik a szükséges oszlopokat (attribútumokat) és a művelet eredményét tartalmazza, a másik meg az elvetett oszlopokat. A dolgozók vezeték- és keresztneveinek, valamint a fizetéseiknek a kilistázásához a projekció műveletét például a következőképpen használhatjuk:

- $\pi_{\text{Vnév, Knév, Fizetés}}(\text{DOLGOZÓ})$

Az eredményül kapott relációt mutatja a 6.1. (b) ábra. A projekció műveletének általános alakja

- $\pi_{\langle \text{attribútumlista} \rangle}(R)$,

ahol a π (a görög pi) a projekció műveletét reprezentáló szimbólum, míg az ⟨attribútumlista⟩ a kívánt attribútumok listája az *R* reláció attribútumai közül. Ismét megemlítendő, hogy *R* általában egy *relációalgebrai kifejezés*, amelynek az eredménye egy reláció, amely a legegyszerűbb esetben mindössze egy adatbázisrelációnak a neve. A projekció műveletének az eredménye csak az ⟨attribútumlista⟩ attribútumaival rendelkezik, mégpedig *abban a sorrendben, ahogyan azok a listában szerepelnek*. Ennélfogva a **foka** egyenlő az ⟨attribútumlista⟩-beli attribútumoknak a számával.

Ha az attribútumlista kizárólag *R* nem kulcs attribútumait tartalmazza, valószínűleg akadnak majd duplikált rekordok. A projekció művelete *eltávolítja a duplikált rekordokat*, így a művelet eredménye egy rekordhalmaz, és ennélfogva egy érvényes reláció lesz. Ezt **duplikáció eliminációnak** nevezik. Tekintsük például a következő projekció műveletet:

- $\pi_{\text{Nem, Fizetés}}(\text{DOLGOZÓ})$

Az eredményt a 6.1. (c) ábra mutatja. Figyeljük meg, hogy a ⟨'F', 325000⟩ rekord csak egyszer jelenik meg a 6.1. (c) ábrán, habár ez az értékkombináció kétszer is előfordul a DOLGOZÓ relációban. A duplikáció elimináció a duplikációk felfedezése érdekében magában foglal egy csoportosítást, és ennélfogva több tennivalót igényel. Ha a duplikációkat nem távolítanánk el, az eredmény rekordoknak egy **multihalmaza** lenne egy halmaz helyett. Ez nem megengedett a formális relációs modellben, de elfogadható a gyakorlatban.

Egy projekció művelet eredményeként kapott relációban a rekordok száma mindig kisebb vagy

egyenlő, mint az *R*-beli rekordok száma. Ha a projekciós lista *R* egy szuperkulcsa — azaz tartalmazza *R* valamelyik kulcsát —, akkor az eredmény relációnak *ugyanannyi* rekordja lesz, mint *R*-nek. Ráadásul

- $\pi_{\langle \text{lista 1} \rangle}(\pi_{\langle \text{lista 2} \rangle}(R) = \pi_{\langle \text{lista 1} \rangle}(R)$,

feltéve hogy ⟨lista 2⟩ tartalmazza a ⟨lista 1⟩-beli attribútumokat; ellenkező esetben a bal oldal egy hibás kifejezés. Figyelemre méltó továbbá, hogy a kommutativitás *nem* teljesül a projekció műveletére.

Műveletsorozatok és az átnevezés művelete

A 6.1. ábrán bemutatott relációknak nincsenek neveik. Általában különféle relációalgebrai műveleteket szeretnénk alkalmazni egymás után. Felírhatjuk a műveleteket egyetlen **relációalgebrai kifejezésként** a műveletek egymásba ágyazásával, vagy megtehetjük, hogy egyszerre csak egy műveletet alkalmazunk, és létrehozhatunk közbenső eredmény relációkat. Ez utóbbi esetben neveket kell adnunk azoknak a relációknak, amelyek a közbenső eredményeket tárolják. Az összes olyan dolgozó vezetéknevének, keresztnevének és fizetésének a lekérdezéséhez például, akik az 5-ös osztályon dolgoznak, egy szelekció és egy projekció műveletet kell alkalmaznunk. Felírhatunk egyetlen relációalgebrai kifejezést a következőképpen:

- $\pi_{\text{Vnév, Knév, Fizetés}}(\sigma_{\text{Osz=5}}(\text{DOLGOZÓ}))$

A 6.2. (a) mutatja ennek a relációalgebrai kifejezésnek az eredményét. Egy másik lehetőségként konkrétan megmutathatjuk a műveletsorozatot is, nevet adva minden egyes közbenső relációnak:

- OSZT5_DOLG ← $\sigma_{\text{Osz=5}}(\text{DOLGOZÓ})$
- EREDMÉNY ← $\pi_{\text{Vnév, Knév, Fizetés}}(\text{OSZT5\_DOLG})$

Gyakran egyszerűbb felbontani egy összetett műveletsorozatot közbenső eredmény relációk megadásával, mint felírni egyetlen relációalgebrai kifejezésként. Ezt a technikát a közbenső és az eredmény relációkban szereplő attribútumok **átnevezésére** is használhatjuk. Ez az olyan összetettebb műveletekkel kapcsolatban lehet hasznos, mint például az egyesítés (unió) vagy az összekapcsolás (join), ahogyan azt majd látni fogjuk. Az attribútumok átnevezéséhez egy relációban egyszerűen csak fel kell sorolnunk az új attribútumneveket zárójelek között, ahogyan azt a következő példában is tesszük:

- TEMP ← $\sigma_{\text{Osz=5}}(\text{DOLGOZÓ})$
- *R*(Vezetéknév, Keresztnév, Fizetés) ← $\pi_{\text{Vnév, Knév, Fizetés}}(\text{TEMP})$

Ezt a két műveletet a 6.2. (b) ábra illusztrálja.

6.2. ábra - Egy műveletsorozat eredménye. (a) $\pi_{\text{Vnév, Knév, Fizetés}}(\sigma_{\text{Osz=5}}(\text{DOLGOZÓ}))$. (b) Közbenső relációk használata és az attribútumok átnevezése.

(a)

Vnév	Knév	Fizetés
Kovács	László	390000
Szabó	Mária	520000
Horváth	Erzsébet	494000
Tóth	János	325000

(b)

TEMP

Vnév	Knév	Szsz	Szdátum	Lakcím	Nem	Fizetés	Főnök_szsz	Osz
Kovács	László	1 650109 0812	1965. január 9.	4033 Debrecen	F	390000	2 551208 2219	5
Szabó	Mária	2 551208 2219	1955. december 8.	1097 Budapest	N	520000	1 371110 4519	5
Horváth	Erzsébet	2 620915 3134	1962. szeptember 15.	1092 Budapest	N	494000	2 551208 2219	5
Tóth	János	1 720731 2985	1972. július 31.	6726 Szeged	F	325000	2 551208 2219	5

R

Vezetéknév	Keresztnév	Fizetés
Kovács	László	390000
Szabó	Mária	520000
Horváth	Erzsébet	494000
Tóth	János	325000

Ha nem alkalmazunk átnevezést, egy szelekció művelet eredményeképpen kapott relációban az attribútumok nevei ugyanazok lesznek, mint az eredeti relációban voltak, ugyanabban a sorrendben. Egy projekció művelet esetén, ha nincs átnevezés, az eredmény relációnak az attribútumnevei a projekciós listában lévőkével lesznek azonosak, és abban a sorrendben fognak szerepelni, ahogyan a listában is megjelennek.

Unáris operátorként definiálhatunk egy formális **átnevezés** műveletet is, amely átnevezi egy reláció nevét vagy az attribútumneveit, vagy esetleg mindegyiket. Az általános átnevezés műveletet, amikor egy *n*-edfokú relációra alkalmazzuk, a következő három alak egyikével jelöljük:

- $\rho_{S(B1, B2, \ldots, Bn)}(R)$ vagy $\rho_S(R)$ vagy $\rho_{(B1, B2, \ldots, Bn)}(R)$,

ahol a ρ (a görög rhó) szimbólumot az átnevezés művelet jelölésére használjuk, *S* az új relációnév, míg $B_1, B_2, \ldots, B_n$ az új attribútumnevek. Az első kifejezés a relációt és az attribútumokat is átnevezi, a második csak a relációt nevezi át, a harmadik pedig csak az attribútumokat. Ha *R* attribútumai $(A_1, A_2, \ldots, A_n)$, ebben a sorrendben, akkor minden egyes A_i B_i-re lesz átnevezve.

Halmazelméleti relációalgebrai műveletek

Az egyesítés (unió), metszet és különbség (kivonás) műveletek

A relációalgebrai műveletek következő csoportját a halmazokon értelmezett hagyományos matematikai műveletek alkotják. Például az összes olyan dolgozó személyi számának a lekérdezéséhez, akik az 5-ös osztályon dolgoznak, vagy akik közvetlen főnökei egy olyan dolgozónak, aki az 5-ös osztályon dolgozik, az egyesítés (unió) műveletet használhatjuk a következőképpen:

- OSZT5_DOLG ← $\sigma_{Osz=5}$(DOLGOZÓ)

- EREDMÉNY1 ← π_{Szsz}(OSZT5_DOLG)
- EREDMÉNY2(Szsz) ← $\pi_{Főnök\_szsz}$(OSZT5_DOLG)
- EREDMÉNY ← EREDMÉNY1 ∪ EREDMÉNY2

Az EREDMÉNY1 reláció tartalmazza az összes olyan dolgozó Szsz-ét, akik az 5-ös osztályon dolgoznak, míg az EREDMÉNY2 reláció az összes olyan dolgozóét, aki közvetlen főnöke egy olyan dolgozónak, aki az 5-ös osztályon dolgozik. Az egyesítés (unió) művelet azokat a rekordokat állítja elő, amelyek vagy az EREDMÉNY1-ben, vagy az EREDMÉNY2-ben, vagy mindkettőben benne vannak (lásd a 6.3. ábrát). Ily módon a ⟨'2 551208 2219'⟩-es Szsz érték csak egyszer jelenik meg az eredményben.

6.3. ábra - Az EREDMÉNY ← EREDMÉNY1 ∪ EREDMÉNY2 egyesítés (unió) művelet eredménye.

EREDMÉNY1

Szsz
1 650109 0812
2 551208 2219
2 620915 3134
1 720731 2985

EREDMÉNY2

Szsz
2 551208 2219
1 371110 4519

EREDMÉNY

Szsz
1 650109 0812
2 551208 2219
2 620915 3134
1 720731 2985
1 371110 4519

Számos halmazelméleti műveletet használunk két halmaz elemeinek különféle módokon történő összefésülésére, többek között az egyesítést (unióképzést), a metszetet és a különbséget (kivonást). Ezek **bináris** műveletek, azaz mindegyiket két halmazra (relációra) alkalmazzuk. Amikor ezeket a műveleteket relációs adatbázisokhoz illesztjük, a két relációnak, amelyekre e három művelet valamelyikét alkalmazzuk, azonos **típusú rekordokkal** kell rendelkeznie; ezt a feltételt hívják *uniókompatibilitásnak*. Az $R(A_1, A_2, \ldots A_n)$ és $S(B_1, B_2, \ldots B_n)$ relációkat **uniókompatibilisnek** mondjuk, ha mindkettő egyformán n-edfokú és ha $\text{dom}(A_i) = \text{dom}(B_i)$ minden $1 \leq i \leq n$ esetén. Ez azt jelenti, hogy a két relációnak azonos számú attribútumának kell lenni, és a megfelelő attribútumpároknak azonos tartománnyal kell rendelkezniük.

Uniókompatibilis R és S relációkra a következőképpen definiálhatjuk az egyesítés (unió), metszet és különbség (kivonás) műveleteket:

- ❒ Egyesítés (unió): Ennek a műveletnek az eredménye, amit $R \cup S$-sel jelölünk, egy olyan reláció, amely magában foglalja az összes olyan rekordot, amely benne van vagy R-ben, vagy S-ben, vagy mindkettőben (R-ben is és S-ben is). A duplikált rekordok eltávolításra kerülnek.
- ❒ Metszet: Ennek a műveletnek az eredménye, amit $R \cap S$-sel jelölünk, egy olyan reláció, amely magában foglalja az összes olyan rekordot, amely benne van R-ben is és S-ben is.
- ❒ Különbség (kivonás): Ennek a műveletnek az eredménye, amit $R - S$-sel jelölünk, egy olyan reláció, amely magában foglalja az összes olyan rekordot, amely benne van R-ben, de nincs benne S-ben.

Megállapodás szerint az eredmény relációnak ugyanazok lesznek az attribútumnevei, mint amik az *első* (R) relációnak voltak. Az átnevezés művelet segítségével később mindig át lehet nevezni az eredménybeli attribútumokat.

A 6.4. ábra bemutatja ezt a három műveletet. A HALLGATÓ és az OKTATÓ a 6.4. (a) ábrán uniókompatibilisek, rekordjaik hallgatók és oktatók neveit szemléltetik. Az egyesítés (unió) művelet eredménye a 6.4. (b) ábrán az összes hallgató és oktató nevét mutatja. Figyeljük meg, hogy a

duplikált rekordok csak egyszer jelennek meg az eredményben. A metszet művelet eredménye (6.4. (c) ábra) csak azokat tartalmazza, akik egyaránt hallgatók és oktatók is.

6.4. ábra - Az egyesítés (unió), metszet és különbség (kivonás) műveletek. (a) Két uniókompatibilis reláció. (b) HALLGATÓ ∪ OKTATÓ. (c) HALLGATÓ ∩ OKTATÓ. (d) HALLGATÓ − OKTATÓ. (e) OKTATÓ − HALLGATÓ.

(a) HALLGATÓ

Vn	Kn
Kovács	László
Szabó	Mária
Kiss	István
Takács	József
Horváth	Erzsébet
Tóth	János
Fazekas	Ilona

OKTATÓ

Vnév	Knév
Nagy	Zoltán
Varga	Gábor
Kovács	László
Dudás	Péter
Szabó	Mária

(b)

Vn	Kn
Kovács	László
Szabó	Mária
Kiss	István
Takács	József
Horváth	Erzsébet
Tóth	János
Fazekas	Ilona
Nagy	Zoltán
Varga	Gábor
Dudás	Péter

(c)

Vn	Kn
Kovács	László
Szabó	Mária

(d)

Vn	Kn
Kiss	Istán
Takács	József
Horváth	Erzsébet
Tóth	János
Fazekas	Ilona

(e)

Vnév	Knév
Nagy	Zoltán
Varga	Gábor
Dudás	Péter

Megjegyzendő, hogy az egyesítés (unió) és metszet műveletek *kommutatívak*, azaz

- $R \cup S = S \cup R$ és $R \cap S = S \cap R$.

Mind az egyesítés (unió), mind a metszet művelet kezelhető tetszőleges számú relációra alkalmazható n-edfokú műveletként, mivel mindkettő *asszociatív művelet*, azaz

- $R \cup (S \cup T) = (R \cup S) \cup T$ és $R \cap (S \cap T) = (R \cap S) \cap T$.

A különbség (kivonás) művelet *nem kommutatív*, azaz általában

- $R - S \neq S - R$.

A 6.4. (d) ábra mutatja azoknak a hallgatóknak a neveit, akik nem oktatók, a 6.4. (e) ábra pedig azokét az oktatókét, akik nem hallgatók.

Figyeljük meg, hogy a metszet kifejezhető az egyesítés (unió) és a különbség segítségével a következőképpen:

- $R \cap S = R \cup S - (R - S) - (S - R)$.

A Descartes-szorzat művelete

A következőkben a — **keresztszorzatként** vagy **kereszt összekapcsolásként** is ismert — **Descartes-szorzat** műveletét tárgyaljuk, amelyet a × műveleti jellel jelölünk. Ez is egy bináris halmazművelet, azonban azoknak a relációknak, amelyekre alkalmazzuk, *nem* kell uniókompatibiliseknek lenniük. Bináris formájában ez a halmazművelet az egyik reláció (halmaz)

minden tagjának (rekordjának) a másik reláció (halmaz) minden tagjával (rekordjával) történő kombinálásával állít elő egy új elemet. Általánosan, $R(A_1, A_2, \ldots, A_n) \times S(B_1, B_2, \ldots, B_m)$ eredménye egy $n + m$ fokú, $Q(A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m)$ sémájú Q reláció, ahol az attribútumok szigorúan ebben a sorrendben szerepelnek. A Q eredmény relációban egy-egy rekord felel meg az R-ből és S-ből származó rekordok minden egyes kombinációjának. Ennélfogva ha R-nek n_R rekordja van (amit úgy jelölünk, hogy $|R| = n_R$), S-nek pedig n_S, akkor $R \times S$-nek $n_R * n_S$ rekordja lesz.

Az n-edfokú Descartes-szorzat művelete a fenti fogalomnak egy kiterjesztése, amely az n darab alaprelációból származó rekordok összes lehetséges kombinációjának a konkatenálásával állít elő új rekordokat. A művelet önmagában történő alkalmazása általában értelmetlen dolog. Akkor hasznos, amikor egy olyan szelekció követi, amely az alaprelációkból származó attribútumok értékeit hasonlítja össze. Tegyük fel például, hogy le szeretnénk kérdezni minden egyes nő dolgozó hozzátartozójának a nevét (természetesen a dolgozó nevével együtt). Ezt a következőképpen tehetjük meg:

- NŐ_DOLG ← $\sigma_{\text{Nem = 'N'}}$(DOLGOZÓ)
- DOLG_NÉV ← $\pi_{\text{Vnév, Knév, Szsz}}$(NŐ_DOLG)
- DOLG_HOZZÁTARTOZÓ ← DOLG_NÉV × HOZZÁTARTOZÓ
- VALÓDI_HOZZÁTARTOZÓ ← $\sigma_{\text{Szsz = Dszsz}}$(DOLG_HOZZÁTARTOZÓ)
- EREDMÉNY ← $\pi_{\text{Vnév, Knév, Hozzátartozó_név}}$(VALÓDI_HOZZÁTARTOZÓ)

Ennek a műveletsorozatnak az eredmény relációi a 6.5. ábrán láthatók. A DOLG_HOZZÁTARTOZÓ reláció a 6.5. ábra DOLG_NÉV relációjára és az 5.4. ábra HOZZÁTARTOZÓ relációjára alkalmazott Descartes-szorzat eredménye. A DOLG_HOZZÁTARTOZÓ relációban a DOLG_NÉV reláció minden egyes rekordja össze van párosítva a HOZZÁTARTOZÓ reláció minden egyes rekordjával, egy nem túl értelmes eredményt előállítva így. Mi egy nő dolgozó rekordját csak a saját hozzátartozóinak a rekordjaival szeretnénk összepárosítani — nevezetesen azokkal a HOZZÁTARTOZÓ rekordokkal, amelyeknek a Dszsz értéke megegyezik a DOLGOZÓ rekord Szsz értékével. A VALÓDI_HOZZÁTARTOZÓ reláció már csak ezeket a rekordokat tartalmazza. A DOLG_HOZZÁTARTOZÓ reláció jó példa arra az esetre, amikor a relációalgebrát helyesen alkalmazva olyan eredményt kapunk, amelynek semmi értelme. Emiatt a felhasználó felelőssége figyelmet fordítani arra, hogy csak értelmes műveleteket hajtson végre a relációkon.

6.5. ábra - A Descartes-szorzat (keresztszorzás) művelet.

A Descartes-szorzat két reláció attribútumainak a kombinációit tartalmazó rekordokat állít elő. A kapcsolódó rekordokat a két relációból egy megfelelő szelekciós feltétellel tudjuk kiválasztani, ahogy az előző példában is csináltuk. Mivel ezt a Descartes-szorzatból és az őt követő szelekcióból álló műveletsorozatot nagyon gyakran használjuk két reláció kapcsolódó rekordjainak az azonosítására és kiválasztására, egy összekapcsolásnak nevezett speciális műveletet hoztak létre ezen műveletsorozat egy műveletként történő megadására. A következőkben az összekapcsolás műveletét tárgyaljuk.

Bináris relációs operátorok: az összekapcsolás (join) és a hányadosképzés

Az összekapcsolás művelete

6.6. ábra - Az OSZTÁLYVEZETŐ ← OSZTÁLY $\bowtie_{\text{Vez\_szsz = Szsz}}$ DOLGOZÓ join művelet eredménye.

OSZTÁLYVEZETŐ

Onév	Oszám	Vez_szsz	...	Vnév	Knév	Szsz	...
Kutatás	5	2 551208 2219	...	Szabó	Mária	2 551208 2219	...
Humán erőforrás	4	2 690329 1099	...	Fazekas	Ilona	2 690329 1099	...
Központ	1	1 371110 4519	...	Nagy	Zoltán	1 371110 4519	...

Az összekapcsolás változatai: az equijoin és a természetes összekapcsolás

A relációalgebrai műveletek teljes halmaza

Az osztás (hányadosképzés) művelete

A lekérdezési fák jelölései

6.1. táblázat - A relációalgebra műveletei

Művelet	Hatás	Jelölés
Szelekció	Kiválaszt minden olyan rekordot az R relációból, amelyek megfelelnek a szelekciós feltételnek.	$\sigma_{\text{szelekciós feltétel}}(R)$
Projekció	Előállít egy új relációt, amely R-nek csak a felsorolt attribútumait tartalmazza, és eltávolítja a duplikált rekordokat.	$\pi_{\text{attribútumlista}}(R)$
Theta join	Előállítja R_1 és R_2 rekordjainak összes olyan kombinációját, amely eleget tesz az összekapcsolási feltételnek.	$R_1 \bowtie_{\text{összekapcsolási feltétel}} R_2$
Equijoin	Előállítja R_1 és R_2 rekordjainak összes olyan kombinációját, amely eleget tesz a kizárólag egyenlőségvizsgálatot tartalmazó összekapcsolási feltételnek.	$R_1 \bowtie_{\text{összekapcsolási feltétel}} R_2$ vagy $R_1 \bowtie_{(\langle\text{összekapcsoló attribútumok 1}\rangle), (\langle\text{összekapcsoló attribútumok 2}\rangle)} R_2$
Természetes összekapcsolás (NATURAL JOIN)	Ugyanaz, mint az equijoin, kivéve hogy az R_2 összekapcsoló attribútumai nem kerülnek be az eredmény relációba; ha az összekapcsoló attribútumoknak azonosak a neveik, akkor egyáltalán nem kell őket megadni.	$R_1 \bowtie_{\text{összekapcsolási feltétel}} R_2$ vagy $R_1 \bowtie_{(\langle\text{összekapcsoló attribútumok 1}\rangle), (\langle\text{összekapcsoló attribútumok 2}\rangle)} R_2$ vagy $R_1 * R_2$
Unió	Előállít egy olyan relációt, amely azokat a rekordokat tartalmazza, amelyek benne vannak	$R_1 \cup R_2$

Művelet	Hatás	Jelölés
	R_1-ben vagy R_2-ben vagy R_1-ben is és R_2-ben is; R_1-nek és R_2-nek uniókompatibilisnek kell lennie.	
Metszet	Előállít egy olyan relációt, amely azokat a rekordokat tartalmazza, amelyek R_1-ben is és R_2-ben is benne vannak; R_1-nek és R_2-nek uniókompatibilisnek kell lennie.	$R_1 \cap R_2$
Különbség	Előállít egy olyan relációt, amely az összes olyan rekordot tartalmazza, amelyek benne vannak R_1-ben, de nincsenek benne R_2-ben; R_1-nek és R_2-nek uniókompatibilisnek kell lennie.	$R_1 - R_2$
Descartes-szorzat	Előállít egy olyan relációt, amely R_1 és R_2 attribútumait tartalmazza, és rekordjai az R_1 és R_2 rekordjainak összes lehetséges kombinációjaként állnak elő.	$R_1 \times R_2$
Osztás (hányadosképzés)	Előállít egy olyan $R(X)$ relációt, amely azokat az $R_1(Z)$-beli $t[X]$ rekordokat tartalmazza, amelyek az összes $R_2(Y)$-beli rekorddal kombinálva előfordulnak R_1-ben, ahol $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

További relációs operátorok

Az általánosított projekció művelete

A külső összekapcsolás műveletei

Példák relációalgebrai lekérdezésekre

A következőkben további példákat adunk a relációalgebrai műveletek használatának illusztrálására. Mindegyik példa az 5.4. ábra adatbázisára hivatkozik. Általában ugyanaz a kérdés számos módon megválaszolható a különféle műveletek felhasználásával. Mi minden kérdést egyféle módon fogunk megválaszolni, és az olvasóra hagyjuk az ekvivalens formulák kitalálását.

1. lekérdezés. Kérdezze le az összes olyan dolgozónak a nevét és a lakcímét, aki a „Kutatás” osztályon dolgozik!

- KUTATÁS_OSZT ← $\sigma_{\text{Onév = 'Kutatás'}}$(OSZTÁLY)
- KUTATÁS_DOLG ← (KUTATÁS_OSZT $\bowtie_{\text{Oszám = Osz}}$ DOLGOZÓ)
- EREDMÉNY ← $\pi_{\text{Vnév, Knév, Lakcím}}$(KUTATÁS_DOLG)

Egyetlen kifejezésként felírva, a lekérdezés az alábbi alakot ölti:

- $\pi_{\text{Vnév, Knév, Lakcím}}(\sigma_{\text{Onév = 'Kutatás'}}(\text{OSZTÁLY} \bowtie_{\text{Oszám = Osz}} \text{DOLGOZÓ}))$

Ezt a lekérdezést meg lehetne adni másképpen is; például az összekapcsolás és a szelekció műveletek sorrendje felcserélhető, vagy az összekapcsoló attribútumok valamelyikének átnevezése után az összekapcsolás helyettesíthető egy természetes összekapcsolással.

2. lekérdezés. Minden kecskeméti projekt esetén listázza ki a projekt számát, a projektet irányító osztály számát, valamint az osztályvezető vezetéknevét, lakcímét és születési dátumát!

- $\text{KECSKEMÉTI\_PROJ} \leftarrow \sigma_{\text{Phelyszín = 'Kecskemét'}}(\text{PROJEKT})$
- $\text{IR\_OSZT} \leftarrow (\text{KECSKEMÉTI\_PROJ} \bowtie_{\text{Osz = Oszám}} \text{OSZTÁLY})$
- $\text{PROJ\_OSZT\_VEZ} \leftarrow (\text{IR\_OSZT} \bowtie_{\text{Vez\_szsz = Szsz}} \text{DOLGOZÓ})$
- $\text{EREDMÉNY} \leftarrow \pi_{\text{Pszám, Oszám, Vnév, Lakcím, Szdátum}}(\text{PROJ\_OSZT\_VEZ})$

3. lekérdezés. Adja meg azoknak a dolgozóknak a vezeték- és keresztnevét, akik *minden* olyan projekten dolgoznak, amit az 5-ös számú osztály irányít!

- $\text{OSZT5\_PROJ(Psz)} \leftarrow \pi_{\text{Pszám}}(\sigma_{\text{Osz = 5}}(\text{PROJEKT}))$
- $\text{DOLG\_PROJ(Szsz, Psz)} \leftarrow \pi_{\text{Dsz, Psz}}(\text{DOLGOZIK\_RAJTA})$
- $\text{E\_DOLG\_SZSZ} \leftarrow \text{DOLG\_PROJ} \div \text{OSZT5\_PROJ}$
- $\text{EREDMÉNY} \leftarrow \pi_{\text{Vnév, Knév}}(\text{E\_DOLG\_SZSZ} * \text{DOLGOZÓ})$

4. lekérdezés. Készítse el azon projektek projektszámainak listáját, amelyekhez köze van „Kovács” vezetéknevű dolgozónak, akár a projekten munkálkodó dolgozóként, akár a projektet irányító osztály vezetőjeként!

- $\text{KOVÁCSOK(Dszsz)} \leftarrow \pi_{\text{Szsz}}(\sigma_{\text{Vnév = 'Kovács'}}(\text{DOLGOZÓ}))$
- $\text{KOVÁCS\_MUNKÁS\_PROJ} \leftarrow \pi_{\text{Psz}}(\text{DOLGOZIK\_RAJTA} * \text{KOVÁCSOK})$
- $\text{VEZETŐK} \leftarrow \pi_{\text{Vnév, Oszám}}(\text{DOLGOZÓ} \bowtie_{\text{Szsz = Vszsz}} \text{OSZTÁLY})$
- $\text{KOVÁCS\_IR\_OSZT(Osz)} \leftarrow \pi_{\text{Oszám}}(\sigma_{\text{Vnév = 'Kovács'}}(\text{VEZETŐK}))$
- $\text{KOVÁCS\_IR\_PROJ} \leftarrow \pi_{\text{Pszám}}(\text{KOVÁCS\_IR\_OSZT} * \text{PROJEKT})$
- $\text{EREDMÉNY} \leftarrow (\text{KOVÁCS\_MUNKÁS\_PROJ} \cup \text{KOVÁCS\_IR\_PROJ})$

Egyetlen kifejezésként felírva, a lekérdezés az alábbi alakot ölti:

- $\pi_{\text{Psz}}(\text{DOLGOZIK\_RAJTA} \bowtie_{\text{Dszsz = Szsz}} (\pi_{\text{Szsz}}(\sigma_{\text{Vnév = 'Kovács'}}(\text{DOLGOZÓ}))) \cup \pi_{\text{Psz}}((\pi_{\text{Oszám}}(\sigma_{\text{Vnév = 'Kovács'}}(\pi_{\text{Vnév, Oszám}}(\text{DOLGOZÓ}))) \bowtie_{\text{Szsz = Vez\_szsz}} \text{OSZTÁLY})) \bowtie_{\text{Oszám = Osz}} \text{PROJEKT})$

5. lekérdezés. Kérdezze le azoknak a dolgozóknak a vezeték- és keresztnevét, akiknek nincs egyetlen hozzátartozójuk sem!

Ez egy olyan típusú lekérdezésre példa, amely a kivonás műveletet használja.

- $\text{ÖSSZES\_DOLG} \leftarrow \pi_{\text{Szsz}}(\text{DOLGOZÓ})$
- $\text{DOLG\_HTVAL(Szsz)} \leftarrow \pi_{\text{Dszsz}}(\text{HOZZÁTARTOZÓ})$
- $\text{DOLG\_HT\_NÉLKÜL} \leftarrow (\text{ÖSSZES\_DOLG} - \text{DOLG\_HTVAL})$

- EREDMÉNY ← $\pi_{\text{Vnév, Knév}}$(DOLG_HT_NÉLKÜL * DOLGOZÓ)

Egyetlen kifejezésként felírva, a lekérdezés az alábbi alakot ölti:

- $\pi_{\text{Vnév, Knév}}$((π_{Szsz}(DOLGOZÓ) − ρ_{Szsz}(π_{Dszsz}(HOZZÁTARTOZÓ))) * DOLGOZÓ)

6. lekérdezés. Listázza ki azoknak az osztályvezetőknek a nevét, akiknek legalább egy hozzátartozójuk van!

- VEZETŐK(Szsz) ← $\pi_{\text{Vez\_szsz}}$(OSZTÁLY)
- DOLG_HTVAL(Szsz) ← π_{Dszsz}(HOZZÁTARTOZÓ)
- VEZETŐ_HTVAL ← (VEZETŐK ∩ DOLG_HTVAL)
- EREDMÉNY ← $\pi_{\text{Vnév, Knév}}$(VEZETŐ_HTVAL * DOLGOZÓ)

Amint azt korábban említettük, ugyanaz a lekérdezés többféle módon is megadható. A műveletek például gyakran különböző sorrendben is alkalmazhatók. Továbbá egyes műveletek helyettesíthetők másokkal; például a metszet művelet a 6. lekérdezésben helyettesíthető egy természetes összekapcsolással. Egy feladatként próbálja meg megcsinálni a fenti példa lekérdezését különböző műveletek felhasználásával![24] Bemutattuk, hogyan lehet egyetlen relációalgbrai kifejezéssel felírni a 1., 4. és 5. lekérdezéseket. Próbálja meg a többi lekérdezést is egyetlen kifejezéssel felírni! A . és . alfejezetekben megmutatjuk, hogyan lehet felírni ezeket a lekérdezéseket más relációs nyelveken.

Rekord alapú relációkalkulus

Rekordváltozók és alaprelációk

Kifejezések és formulák a rekord alapú relációkalkulusban

Az egzisztenciális és univerzális kvantorok

Példák az egzisztenciális kvantor használatára

Az univerzális és egzisztenciális kvantorok közötti transzformációk

A következőkben bemutatunk néhány, a matematikai logikából jól ismert transzformációt, amelyek az univerzális és az egzisztenciális kvantorokkal vannak összefüggésben. Egy univerzális kvantort át lehet alakítani egzisztenciális kvantorrá, és ugyanúgy vissza is úgy, hogy ekvivalens kifejezést kapjunk. Egy általános transzformációt informálisan a következőképpen írhatunk le: Alakítsuk át az egyik típusú kvantort a másik típusúvá negációval (**NOT**-ot eléírva); az **AND**-et és az **OR**-t cseréljük fel egymással; a negált formulák legyenek nemnegáltak; a nemnegált formulák pedig negáltak. Az alábbiakban kifejtjük e transzformáció néhány speciális esetét, a ≡ szimbólum az **ekvivalens** fogalmat rövidíti:

- (∀x) (P(x)) ≡ **NOT** (∃x) (**NOT** (P(x)))
- (∃x) (P(x)) ≡ **NOT** (∀x) (**NOT** (P(x)))
- (∀x) (P(x) **AND** Q(x)) ≡ **NOT** (∃x) (**NOT** (P(x)) **OR NOT** (Q(x)))
- (∀x) (P(x) **OR** Q(x)) ≡ **NOT** (∃x) (**NOT** (P(x)) **AND NOT** (Q(x)))

- $(\exists x)\ (P(x)$ **OR** $Q(x)) \equiv$ **NOT** $(\forall x)\ ($**NOT** $(P(x))$ **AND NOT** $(Q(x)))$
- $(\exists x)\ (P(x)$ **AND** $Q(x)) \equiv$ **NOT** $(\forall x)\ ($**NOT** $(P(x))$ **OR NOT** $(Q(x)))$

Megjegyzendő még, hogy igazak a következők, amennyiben a → szimbólum az **implikációt** jelöli:

- $(\forall x)\ (P(x)) \rightarrow (\exists x)\ (P(x))$
- **NOT** $(\exists x)\ (P(x)) \rightarrow$ **NOT** $(\forall x)\ (P(x))$

Az univerzális kvantor használata

Biztonságos kifejezések

Amennyiben univerzális kvantorokat, egzisztenciális kvantorokat vagy predikátumok negációját használjuk egy relációkalkulusbeli kifejezésben, meg kell győződnünk arról, hogy az eredményül kapott kifejezésnek van értelme. A relációkalkulusban **biztonságos kifejezésnek** nevezzük az olyan kifejezést, amely garantáltan *véges számú rekordot* ad eredményül; különben **nem biztonságos kifejezésről** beszélünk. Például a

- { t | **NOT** (DOLGOZÓ(t)) }

kifejezés *nem biztonságos*, mivel az univerzum összes olyan rekordját adja eredményül, amely *nem* DOLGOZÓ rekord, és ezek számossága végtelen. Ha követjük a korábban, a Q3 lekérdezésnél leírt szabályokat, biztonságos kifejezést kapunk, amikor univerzális kvantorokat használunk. Pontosabban is definiálhatjuk a biztonságos kifejezéseket, ha bevezetjük *a rekord alapú relációkalkulusbeli kifejezés tartománya* fogalmat: ez az összes olyan érték halmaza, amely vagy konstans értékként fordul elő a kifejezésben, vagy a kifejezésben hivatkozott relációk bármely rekordjában szerepel. A { t | **NOT** (DOLGOZÓ(t)) } tartománya a DOLGOZÓ reláció összes rekordjában előforduló attribútumértékek halmaza (bármely attribútum esetén). A Q3A kifejezés tartománya magában foglalja az összes olyan értéket, amely előfordul a DOLGOZÓ, a PROJEKT és a DOLGOZIK_RAJTA relációkban (hozzávéve az 5-ös értéket, amely magában a lekérdezésben jelenik meg).

Egy kifejezést **biztonságosnak** nevezünk, ha az eredményében szereplő értékek a kifejezés tartományából származnak. Figyeljük meg, hogy a { t | **NOT** (DOLGOZÓ(t)) } nem biztonságos, mert az eredményében a DOLGOZÓ reláción kívüli rekordokat (és ezért értékeket) tartalmaz; ezek az értékek nincsenek a kifejezés tartományában. Az összes többi példánk biztonságos kifejezés.

Tartomány alapú relációkalkulus

Létezik egy másik fajta relációkalkulus, amit **tartomány alapú relációkalkulusnak** vagy röviden tartománykalkulusnak nevezünk. Míg az SQL-t a rekord alapú relációkalkulusra alapozva fejlesztette ki az IBM Research San Jose-ban, Kaliforniában, addig a QBE-nek (Query-By-Example) nevezett másik nyelvet, ami a tartománykalkulushoz kötődik, ezzel majdnem párhuzamosan fejlesztették az IBM T. J. Watson Research Centerben Yorktown Heights-ban, New York államban. A tartománykalkulus formális specifikációját a QBE rendszer fejlesztése után írták le.

A tartománykalkulus a formulákban használt *változók típusában* különbözik a rekordkalkulustól: a változók ahelyett, hogy a rekordok fölött futnának végig, az attribútumok tartományainak egyszerű értékeit járják be. Ahhoz, hogy kialakítsunk egy n-edfokú relációt egy lekérdezéshez, szükségünk van n ilyen **tartományváltozóra** — mindegyik attribútumhoz egyre. Egy tartománykalkulusbeli kifejezés

- $\{ x_1, x_2, \ldots, x_n \mid \text{FELTÉTEL}(x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}) \}$

alakú, ahol $x_1, x_2, \ldots, x_n, x_{n+1}, x_{n+2}, \ldots, x_{n+m}$ tartományváltozók, amelyek végigfutnak a tartományok (az attribútumok tartományai) fölött, míg a FELTÉTEL a tartomány alapú relációkalkulus egy **feltétele** vagy **formulája**.

Egy formula **atomokból** épül föl. Egy formula atomjai kicsit különböznek a rekordkalkulusban megadottaktól, a következők lehetnek:

- Az $R(x_1, x_2, \ldots x_j)$ alakú atom, ahol R egy j fokú relációnak a neve, és minden x_i egy tartományváltozó ($1 \leq i \leq j$). Ez az atom azt fejezi ki, hogy az $\langle x_1, x_2, \ldots x_j \rangle$ értékek listája egy rekord abban a relációban, amelynek R a neve, és x_i a rekord i-edik attribútumának az értéke. Egy tartománykalkulusbeli kifejezés tömörebb megadása érdekében *elhagyhatjuk a vesszőket* a változók listájában; így használhatjuk a
  - $\{ x_1, x_2, \ldots, x_n \mid R(x_1\ x_2\ \ldots\ x_n)$ **AND** $\ldots \}$

  alakú kifejezést a
  - $\{ x_1, x_2, \ldots, x_n \mid R(x_1, x_2, \ldots, x_n)$ **AND** $\ldots \}$

  helyett.
- Az x_i **op** x_j alakú atom, ahol **op** egyike a $\{=, <, \leq, >, \geq, \neq\}$ halmaz összehasonlító operátorainak, míg x_i és x_j tartományváltozók.
- Az x_i **op** c vagy c **op** x_j alakú atom, ahol **op** egyike a $\{=, <, \leq, >, \geq, \neq\}$ halmaz összehasonlító operátorainak, x_i és x_j tartományváltozók, míg c egy konstans érték.

Ahogyan a rekordkalkulusban is, az atomok IGAZ vagy HAMIS értéket vesznek fel egy adott értékhalmaz esetén, amit az atomok **igazságértékének** nevezünk. Az 1. esetben, ha a tartományváltozókhoz hozzárendelt értékek megfelelnek a megadott R reláció egy rekordjának, akkor az atom értéke IGAZ. A 2. és 3. esetben, ha a tartományváltozókhoz hozzárendelt értékek kielégítik a feltételt, akkor az atom IGAZ.

A rekord alapú relációkalkulushoz hasonlóan a formulák itt is atomokból, változókból és kvantorokból épülnek föl, így most nem fogjuk elismételni a formulákra vonatkozó előírásokat. Következzen inkább néhány, tartománykalkulusban megadott lekérdezés. A tartományváltozókat az ábécé kisbetűivel, az l, m, n, …, x, y, z betűkkel fogjuk jelölni.

0. lekérdezés. Listázza ki azoknak a dolgozóknak a születési dátumát és lakcímét, akiknek a neve „Kovács János”!

Q0: $\{tu \mid (\exists q)\ (\exists r)\ (\exists s)\ (\exists v)\ (\exists w)\ (\exists x)\ (\exists y)$ (DOLGOZÓ($qrstuvwxy$) **AND** q='Kovács' **AND** r='János')$\}$

Kilenc változóra van szükségünk a DOLGOZÓ relációhoz, amelyek végigfutnak az egyes attribútumok tartományai fölött. A kilenc változóból (q, r, s, …, y) csak a t és az u szabadok. Először felsoroljuk a *kért attribútumokat*, a Szdátumot és a Lakcímet a szabad változókkal, t-vel a Szdátumra, u-val a Lakcímre hivatkozva. Azután a függőleges elválasztó vonalat (|) követően megadjuk a feltételt egy rekord kiválasztására — nevezetesen, hogy a *qrstuvwxy* változókhoz rendelt értéksorozatnak a DOLGOZÓ reláció egy rekordjának kell lennie, valamint hogy a q (VNév) és az r (Knév) értéke rendre 'Kovács' és 'János' legyen. A kényelem kedvéért a további példáinkban csak azokat a változókat fogjuk kvantálni, amelyek *ténylegesen megjelennek egy feltételben* (ezek a q és az r lennének a Q0-ban).

Egy alternatív, QBE-ben használt rövidítési lehetőség ennek a lekérdezésnek a felírására a 'Kovács' és 'János' konstansok közvetlen hozzárendelése, ahogyan azt a Q0A-ban láthatjuk. Itt azok a változók, amelyek nem jelennek meg a függőleges vonal bal oldalán, implicit módon

egzisztenciálisan kvantáltak:[25]

Q0A: {*tu* | DOLGOZÓ('Kovács', 'János', *s*, *t*, *u*, *v*, *w*, *x*, *y*) }
1. lekérdezés. Kérdezze le az összes olyan dolgozónak a nevét és a lakcímét, aki a „Kutatás” osztályon dolgozik!

Q1: {*qru* | (∃*y*) (∃*l*) (∃*m*) (DOLGOZÓ(*qrstuvwxy*) **AND** OSZTÁLY(*lmno*) **AND** *l*='Kutatás' **AND** *m*=*y*)}
Az olyan feltétel, mint a *m*=*y* a Q1-ben, amely két olyan tartományváltozót kapcsol össze, amelyek két reláció attribútumai fölött futnak végig, **összekapcsoló** vagy **join feltétel**; míg az olyan feltétel, mint a *l*='Kutatás', amely egy tartományváltozót egy konstanssal kapcsol össze, **szelekciós feltétel**.

2. lekérdezés. Minden kecskeméti projekt esetén listázza ki a projekt számát, a projektet irányító osztály számát, valamint az osztályvezető vezetéknevét, születési dátumát és lakcímét!

Q2: {*ikrtu* | (∃*j*) (∃*m*) (∃*n*) (∃*s*) (PROJEKT(*hijk*) **AND** DOLGOZÓ(*qrstuvwxy*) **AND** OSZTÁLY(*lmno*) **AND** *k*=*m* **AND** *n*=*s* **AND** *j*='Kecskemét')}
6. lekérdezés. Listázza ki azoknak a dolgozóknak a vezeték- és keresztnevét, akiknek nincs egyetlen hozzátartozójuk sem!

Q6: {*qr* | (∃*s*)(DOLGOZÓ(*qrstuvwxy*) **AND** (**NOT**(∃*l*)(HOZZÁTARTOZÓ(*lmnop*) **AND** *s*=*l*)))}
Q6 átírható az egzisztenciális kvantor helyett az univerzális kvantort használva, ahogyan az a Q6A-ban látható:

Q6A: {*qr* | (∃*s*)(DOLGOZÓ(*qrstuvwxy*) **AND** ((∀*l*)(**NOT**(HOZZÁTARTOZÓ(*lmnop*)) **OR** **NOT**(*s*=*l*))))}
7. lekérdezés. Listázza ki azoknak az osztályvezetőknek a nevét, akiknek legalább egy hozzátartozójuk van!

Q7: {*qr* | (∃*s*) (∃*j*) (∃*l*) (DOLGOZÓ(*qrstuvwxy*) **AND** OSZTÁLY(*hijk*) **AND** HOZZÁTARTOZÓ(*lmnop*) **AND** *s*=*j* **AND** *l*=*s*)}
Amint azt korábban már említettük, megmutatható, hogy bármely lekérdezés, amely felírható a relációalgebrában, felírható tartomány alapú vagy rekord alapú relációkalkulusban is. Bármely tartomány alapú vagy rekord alapú relációkalkulusbeli *biztonságos kifejezés* ugyancsak felírható a relációalgebrában.

A QBE nyelv a tartomány alapú relációkalkuluson alapul, bár ezt később valósították meg, a tartománykalkulus formalizálása után. A QBE volt az első, adatbázisrendszerekhez kifejlesztett grafikus lekérdező nyelvek egyike, minimális szintaktikával. Az IBM Research-nél fejlesztették, és hozzáférhető az IBM kereskedelmi termékeként mint a DB2-höz interfészt kínáló Query Management Facility (QMF) része. Sok másik kereskedelmi termék is merített ötleteket belőle, fontos helyet foglal el a relációs nyelvek palettáján.

Összefoglalás

Ebben a fejezetben a relációs adatmodell két formális nyelvét mutattuk be. Ezeket arra használjuk, hogy relációkat kezeljünk, és a lekérdezésekre adott válaszként új relációkat állítsunk elő. Ismertettük a relációalgebrát és a műveleteit, amelyek egy műveletsorozat megadásával adnak választ a lekérdezésre. Ezután a relációkalkulus két típusát mutattuk be: a rekord alapú kalkulust és a tartomány alapú kalkulust, amelyek deklaratívak abban az értelemben, hogy a lekérdezés eredményét anélkül határozzák meg, hogy meg kellene adnunk, hogy az hogyan álljon elő.

A . alfejezettől a . alfejezetig bemutattuk az alapvető relációalgebrai műveleteket, valamint azokat a lekérdezéstípusokat, amelyekben ezek használhatók. Először a szelekció és projekció unáris relációs műveleteket, valamint az átnevezés műveletét tárgyaltuk. Ezután a bináris halmazelméleti műveleteket ismertettük, amelyek megkövetelik, hogy azok a relációk, amelyekre alkalmazzuk

őket, uniókompatibilisek legyenek; ezek az unió, a metszet és a (halmaz)különbség. A Descartes-szorzat művelet olyan halmazművelet, amelynek segítségével két reláció rekordjait tudjuk párosítani az összes lehetséges kombináció előállításával. A gyakorlatban ritkán használjuk; megmutattuk azonban, hogy a Descartes-szorzat egy azt követő szelekcióval hogyan használható két reláció összeillő rekordjainak előállítására, amely az összekapcsolás műveletéhez vezet. Különböző összekapcsolás műveleteket (theta join, equijoin és természetes összekapcsolás) mutattunk be. A relációalgebrai lekérdezések belső reprezentálására bevezettük a lekérdezési fákat.

Bemutattunk néhány fontos lekérdezéstípust, amelyek *nem* fejezhetők ki az alapvető relációalgebrai műveletekkel, de a gyakorlatban fontosak. Ismertettük az általánosított projekció műveletét, ahol a projekciós listában az attribútumok függvényeit is használhatjuk, valamint a csoportosító függvényeket, amelyekkel csoportokra vonatkozó lekérdezéseket tudunk kezelni. Szót ejtettünk a rekurzív lekérdezésekről, amelyeket nem támogat közvetlenül az algebra, de amelyek lépésenkénti megközelítéssel kezelhetők, ahogy azt láttuk. Ismertettük a külső összekapcsolás és a külső unió műveleteket, amelyek kiterjesztik az összekapcsolás és az unió műveleteket, és lehetővé teszik, hogy az eredmény megőrizze a forrás relációkban tárolt összes információt.

Az utolsó két alfejezet a relációkalkulus mögött meghúzódó alapfogalmakat tárgyalta. A relációkalkulus a matematikai logika egyik ágán, a predikátumkalkuluson alapul. Két típusát különböztetjük meg: (1) a rekord alapú relációkalkulust, amely rekordváltozókat használ, ami a relációk rekordjain (sorain) fut végig, és (2) a tartomány alapú relációkalkulust, amely tartományváltozókat használ, amik a tartományokat (a reláció oszlopait) járják be. A relációkalkulusban a lekérdezéseket egyetlen deklaratív utasítással adjuk meg anélkül, hogy a lekérdezés eredményének meghatározására bármilyen módszert vagy műveletsorozatot megadnánk. Emiatt a relációkalkulust gyakran magasabb szintű nyelvnek tekintjük, mint a relációalgebrát, hiszen a relációkalkulusbeli kifejezés azt mondja meg, hogy *mit* akarunk lekérdezni, tekintet nélkül arra, hogy *hogyan* hajtódik végre a lekérdezés.

Megadtuk a relációkalkulusbeli lekérdezések szintaxisát mind rekord-, mind tartományváltozók használatával. Bemutattuk a lekérdezési gráfokat mint a relációkalkulusban felírt lekérdezések belső reprezentációját. Szintén tárgyaltunk az egzisztenciális kvantorról (∃) és az univerzális kvantorról (∀). Láttuk, hogy a relációkalkulusbeli változók kötötté válnak ezen kvantorok használatával. Részletesen ismertettük, hogyan kell univerzális kvantort tartalmazó lekérdezéseket írni, és felvetettük a véges eredményt szolgáltató biztonságos lekérdezések definiálásának problémáját. Bemutattuk az univerzális kvantorok egzisztenciális kvantorrokká történő átalakításának, valamint az ellentétes irányú átalakításnak a szabályait. A kvantorok nagyban megnövelik a relációkalkulus kifejezőerejét, és általuk a relációkalkulus ekvivalenssé válik a relációalgebrával. Az alap relációkalkulusban nincs megfelelője a csoportosításnak és az összesítő függvényeknek, bár már születtek javaslatok ilyen irányú kiterjesztésekre.

Áttekintő kérdések

1. Sorolja fel a relációalgebrai operátorokat, és definiálja működésüket!
2. Mi az uniókompatibilitás? Miért kell azoknak a relációknak, amelyeknek az unióját, a metszetét és a különbségét képezzük, uniókompatibiliseknek lenniük?
3. Soroljon fel néhány olyan kérdésfajtát, amelyeknél mindenképpen szükséges az attribútumok átnevezése ahhoz, hogy a lekérdezést egyértelműen adhassuk meg!
4. Mutassa be a *belső összekapcsolás* műveletének a különböző típusait! Miért van szükség a theta joinra?
5. Milyen szerepet játszik a *külső kulcs* fogalma az életszerű összekapcsolási műveletek legelterjedtebb típusainak megadásakor?
6. Miben különbözik a külső összekapcsolás a belső összekapcsolástól? Miben különbözik a

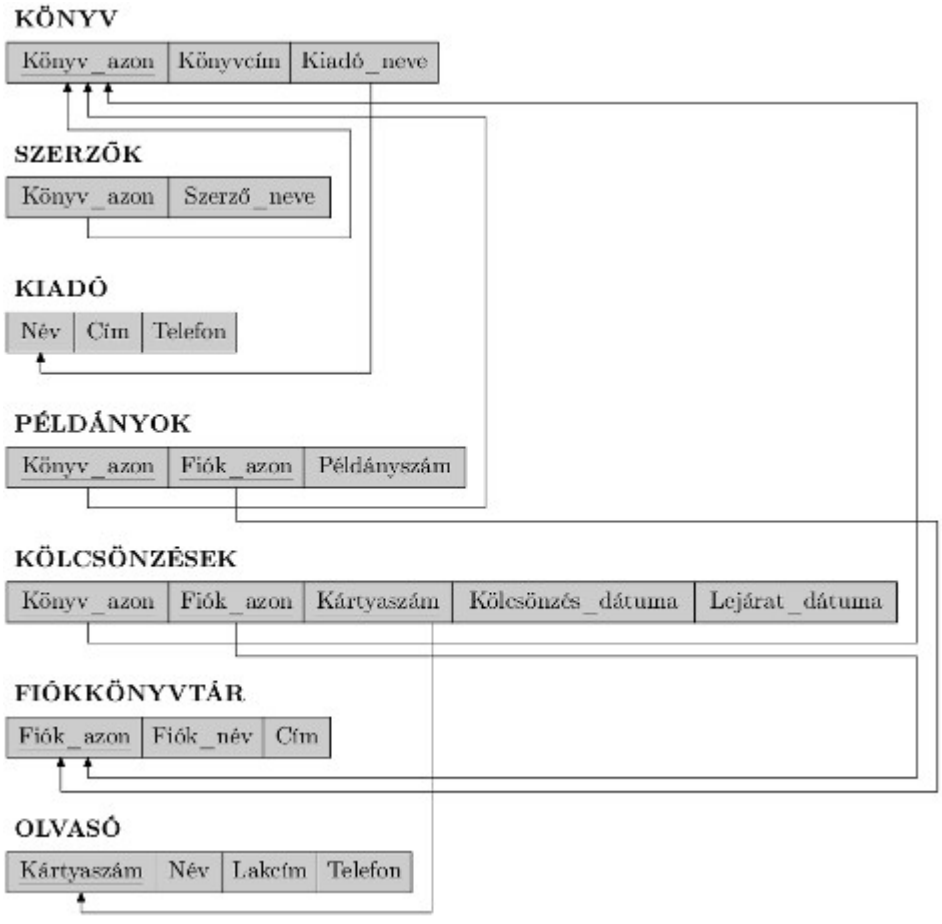
külső unió az uniótól?

7. Miben különbözik a relációkalkulus a relációalgebrától, és miben hasonlítanak egymásra?
8. Miben különbözik a rekord alapú relációkalkulus a tartomány alapú relációkalkulustól?
9. Írja le az egzisztenciális kvantor (∃) és az univerzális kvantor (∀) jelentését!
10. Definiálja az alábbi, rekord alapú relációkalkulushoz kapcsolódó fogalmakat: *rekordváltozó*, *alapreláció*, *atomi formula*, *formula* és *kifejezés*!
11. Definiálja az alábbi, tartomány alapú relációkalkulushoz kapcsolódó fogalmakat: *tartományváltozó*, *alapreláció*, *atomi formula*, *formula* és *kifejezés*!
12. Mit értünk *biztonságos kifejezés* alatt a relációkalkulusban?
13. Mikor nevezünk egy lekérdezőnyelvet relációs értelemben teljesnek?

Feladatok

1. Adja meg a . alfejezetben felírt példa lekérdezések eredményeit, ha azokat a 5.4. ábrán látható adatbázis-állapotra alkalmazzuk!
2. Tekintse a 6.7. ábrán látható KÖNYVTÁR relációs adatbázissémát, amelyet könyvek, olvasók és kölcsönzések nyomon követésére használhatunk. A hivatkozási integritási megszorításokat — az 5.5. ábra jelöléseihez hasonlóan — irányított élek jelzik a 6.7. ábrán.

   6.7. ábra - Relációs adatbázisséma egy KÖNYVTÁR adatbázishoz.



   Írjon relációs kifejezéseket a következő lekérdezésekhez:

   a. *Az elveszett törzs* című könyvnek hány példányával rendelkezik a „Móra Ferenc”

fiókkönyvtár?

b. *Az elveszett törzs* című könyvnek hány példányával rendelkeznek az egyes fiókkönyvtárak?

c. Kérdezze le az összes olyan olvasó nevét, akinél nincsen egyetlen kikölcsönzött könyv sem!

d. Az összes olyan könyv esetén, amelyet a „Móra Ferenc” fiókkönyvtárból kölcsönöztek ki, és amelyeknél a Lejárat_dátuma a mai nap, kérdezze le a könyv címét, valamint az olvasó nevét és lakcímét!

e. Minden egyes fiókkönyvtár esetén kérdezze le a fiókkönyvtár nevét és azoknak a könyveknek a darabszámát, amelyeket onnan (abból a fiókkönyvtárból) kölcsönöztek ki!

f. Kérdezze le minden olyan olvasó nevét, lakcímét és kikölcsönzött könyveinek a darabszámát, akik ötnél több könyvet kölcsönöztek ki!

g. Minden olyan könyv esetén, amelynek a szerzője (vagy társszerzője) Stephen King, kérdezze le a címet és azt, hogy hány példánnyal rendelkezik belőle a „Központi Könyvtár” nevű fiókkönyvtár!

3. Mutassa meg, hogy hogyan adhatók meg a következő relációalgebrai műveletek a rekord alapú, illetve a tartomány alapú relációkalkulusban!

a. $\sigma_{A=C}(R(A, B, C))$

b. $\pi_{\langle A, C\rangle}(R(A, B, C))$

c. $R(A, B, C) * S(C, D, E)$

d. $R(A, B, C) \cup S(A, B, C)$

e. $R(A, B, C) \cap S(A, B, C)$

f. $R(A, B, C) - S(A, B, C)$

g. $R(A, B, C) \times S(D, E, F)$

h. $R(A, B) \div S(A)$

4. Döntse el, hogy igazak-e vagy sem az alábbi következtetések:

a. **NOT** $(P(x)$ **OR** $Q(x)) \rightarrow$ (**NOT** $(P(x))$) **AND** (**NOT** $(Q(x))$)

b. **NOT** $(\exists x)\ (P(x)) \rightarrow \forall x$ (**NOT** $(P(x))$)

c. $(\exists x)\ (P(x)) \rightarrow \forall x\ (P(x))$

[23] Az SQL a rekord alapú relációkalkulusra épül, de a relációalgebrából és azok kiterjesztéseiből is vesz át bizonyos műveleteket.

[24] A lekérdezés-optimalizálás során a rendszer kiválaszt egy olyan műveletsorozatot, amellyel a lekérdezés a lehető leghatékonyabban hajtódik végre.

[25] Megjegyzendő, hogy a csak a feltételekben aktuálisan használt tartományváltozók kvantálása, illetve egy olyan, a tartományváltozók elválasztására szolgáló vesszők nélküli predikátum használata, mint a DOLGOZÓ(*qrstuvwxy*), kizárólag csak kényelmi célból használt rövidítés; semmiképpen sem helyes formális jelölés.

7. fejezet - Relációs adatbázis-tervezés ER és EER sémák relációs sémára történő leképezésével

Tartalom

Ebben a fejezetben azt tárgyaljuk, hogyan lehet egy **relációs adatbázissémát megtervezni** egy koncepcionális sématerv alapján. Ez megfelel a logikai adatbázistervezésnek vagy a . alfejezetben ismertetettt adatmodell-leképezési lépésnek (lásd a 3.1. ábrát). Bemutatjuk azokat az eljárásokat, amelyekkel egy egyed-kapcsolat (ER) vagy egy kibővített egyed-kapcsolat (EER) sémából relációs sémát tudunk készíteni, és összekapcsoljuk a 3. és 4. fejezetben bemutatott ER és EER modellbeli konstrukciókat az 5. és 6. fejezetben bemutatott relációs modellbeli konstrukciókkal. Számos számítógéppel támogatott szoftverfejlesztési (CASE) eszköz az ER, az EER vagy más hasonló modellen alapul, ahogy ezt a 3. és 4. fejezetekben tárgyaltuk. Ezeket az automatizált eszközöket az adatbázis-tervezők interaktívan használják az adatbázis alkalmazások ER vagy EER sémájának kifejlesztésére. Sok ilyen eszköz ER vagy EER diagramokat, vagy ezek valamilyen változatait használja a séma grafikus tervezésére, majd automatikusan átkonvertálja azt relációs adatbázissémára egy konkrét relációs DBMS DDL-jében felírva olyan algoritmusok segítségével, amelyeket ebben a fejezetben ismertetünk.

A . alfejezetben egy hétlépéses algoritmust vázolunk az alapvető ER modellbeli konstrukciók — egyedtípusok (erős és gyenge), bináris kapcsolatok (különböző strukturális megszorításokkal), *n*-edfokú kapcsolatok, valamint attribútumok (egyszerű, összetett és többértékű) — relációsémákra történő konvertálásához. Ezután a . alfejezetben folytatjuk a leképezési algoritmust azzal, hogy leírjuk, hogyan lehet az EER modellbeli konstrukciókat — specializáció/generalizáció és unió típusok (kategóriák) — relációsémákra leképezni.

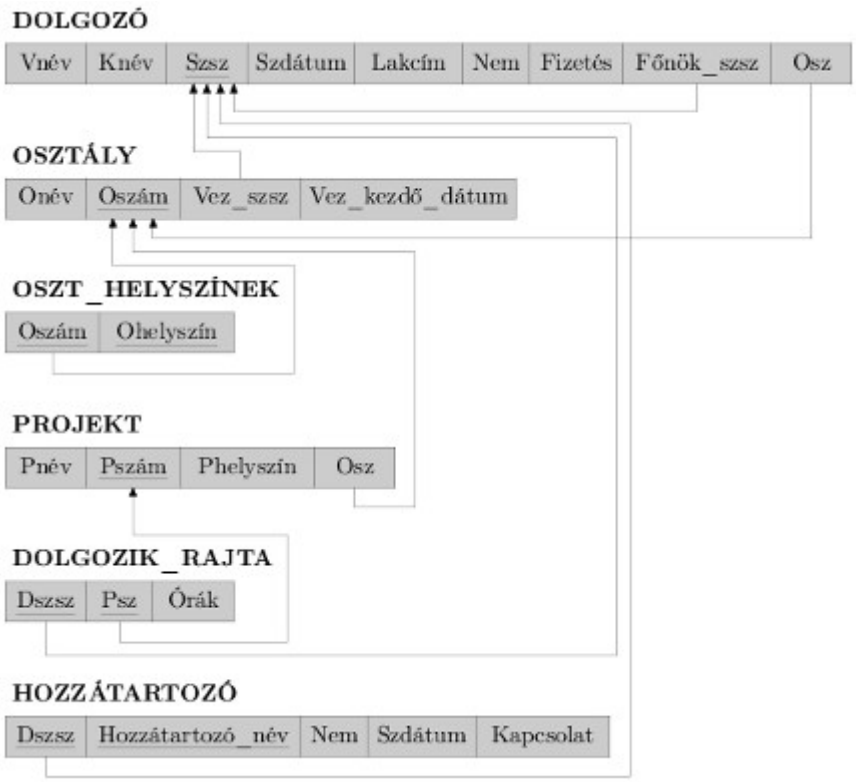
Relációs adatbázis-tervezés az ER séma relációs sémára történő leképezésével

Az ER séma relációs sémára történő leképezésének algoritmusa

Az alábbiakban megadjuk az ER séma relációs sémára történő leképezésének egy algoritmusát. A leképezési eljárás illusztrálására a VÁLLALAT adatbázis példát használjuk. A leképezési lépések illusztrálásához a VÁLLALAT ER sémája a 7.1. ábrán látható ismét, míg a hozzá tartozó VÁLLALAT adatbázissémát a 7.2. ábra mutatja.

7.1. ábra - ER koncepcionális séma a VÁLLALAT adatbázishoz

7.2. ábra - A VÁLLALAT ER séma relációs adatbázissémára történő leképezésének eredménye



1. **Erős egyedtípusok leképezése.** Az ER séma minden *E* erős egyedtípusához rendeljünk hozzá egy *R* relációsémát, amely tartalmazza *E* összes egyszerű attribútumát. Az összetett attribútumoknak csak az egyszerű komponenseit adjuk hozzá *R* attribútumaihoz. Válasszuk *E* kulcs attribútumainak egyikét az *R* relációséma elsődleges kulcsául. Ha az *E*-ből választott kulcs összetett, akkor annak egyszerű attribútumai együttesen fogják alkotni *R* elsődleges kulcsát.

   Ha a koncepcionális tervezés során az *E*-hez több kulcsot is találtunk, akkor az egyes további kulcsokat alkkotó attribútumokat leíró információkat is megtartjuk, hogy az *R* relációséma másodlagos (egyedi) kulcsait definiáljuk velük. A kulcsokról szerzett ismeretek indexelési és más típusú elemzési célból is felhasználhatók.

   A példánkban létrehozzuk a 7.2. ábra DOLGOZÓ, OSZTÁLY és PROJEKT relációsémáit, amelyek a 7.1. ábra DOLGOZÓ, OSZTÁLY és PROJEKT erős egyedtípusainak felelnek meg. A külső kulcs attribútumokat, illetve a kapcsolóattribútumokat, ha vannak is, még nem vesszük hozzájuk, majd csak a későbbi lépésekben. Ilyen attribútum a DOLGOZÓ Főnök_szsz és Osz, az OSZTÁLY Vez_szsz és Vez_kezdő_dátum, továbbá a PROJEKT Osz attribútuma. A példánkban sorban az Szsz-t, az Oszámot és a Pszámot választjuk elsődleges kulcsnak a DOLGOZÓ, az OSZTÁLY és a PROJEKT relációsémákhoz. Azt az ismeretet, hogy az OSZTÁLY Onév és a PROJEKT Pnév attribútuma másodlagos kulcs, megjegyezzük mint a tervezésben később felhasználható lehetőségeket.

   Azokat a relációsémákat, amelyeket az egyedtípusok leképezésével hoztunk létre, néha **egyedtípusból képzett relációsémáknak** is nevezzük, mert relációpéldányaik minden rekordja egy-egy egyed-előfordulást reprezentál. A leképezési lépés utáni eredmény a 7.3. (a) ábrán látható.

   7.3. ábra - A leképezés néhány lépésének bemutatása. (a) *Egyedtípusból képzett* relációsémák az első lépés után. (b) Egy új, *gyenge egyedtípusból képzett* relációséma a

második lépés után. (c) *Kapcsoló* relációséma az 5. lépést követően. (d) Többértékű attribútumot reprezentáló relációséma a 6. lépés után.

(a) **DOLGOZÓ**

Vnév	Knév	Szsz	Szdátum	Lakcím	Nem	Fizetés

OSZTÁLY

Onév	Oszám

PROJEKT

Pnév	Pszám	Phelyszín

(b) **HOZZÁTARTOZÓ**

Dszsz	Hozzátartozó_név	Nem	Szdátum	Kapcsolat

(c) **DOLGOZIK_RAJTA**

Dszsz	Psz	Órák

(d) **OSZT_HELYSZÍNEK**

Oszám	Ohelyszín

2. **Gyenge egyedtípusok leképezése.** Az ER séma minden *W* gyenge egyedtípusához rendeljünk hozzá egy *R* relációsémát, melynek attribútumai legyenek *W* összes egyszerű attribútuma és *W* összetett attribútumainak egyszerű komponensei. Továbbá adjuk hozzá *R* attribútumaihoz külső kulcs attribútumként azoknak a relációsémáknak az elsődleges kulcs attribútumait, amelyeket a domináns egyedtípusoknak feleltettünk meg; ezzel képezzük le a *W*-hez tartozó azonosító kapcsolattípust. *R* elsődleges kulcsa a tulajdonos egyedtípusok elsődleges kulcsainak és a *W* gyenge egyedtípus diszkriminátorának az együttese.

   Ha egy E_2 gyenge egyedtípus tulajdonosa a szintén gyenge E_1 egyedtípus, akkor E_1-et E_2 előtt kell leképezni, mert az E_2 leképezéséhez szükség van az E_1-ből képzett relációséma elsődleges kulcsára.

   A példánkban ebben a lépésben létrehozzuk a HOZZÁTARTOZÓ relációsémát, amely a HOZZÁTARTOZÓ gyenge egyedtípusnak felel meg (lásd a 7.3. (b) ábrát). Felvesszük a DOLGOZÓ relációséma Szsz elsődleges kulcsát — amely a tulajdonos egyedtípusnak felel meg — a HOZZÁTARTOZÓ külső kulcs attribútumaként, és átnevezzük Dszsz-re, bár ezt nem szükséges megtenni. A HOZZÁTARTOZÓ relációséma elsődleges kulcsa a {Dszsz, Hozzátartozó_név} kombináció, mert a Hozzátartozó_név (ami a 7.1. ábra Név attribútumából szintén át lett nevezve) a HOZZÁTARTOZÓ részleges kulcsa.

   Általában a CASCADE opciót szoktuk megadni a hivatkozó táblában bekövetkező eseményre a gyenge egyedtípusnak megfelelő relációséma külső kulcsának definiálásakor, mivel a gyenge egyed létezésfüggően kapcsolódik a tulajdonos egyedéhez. Ez az opció mind ON UPDATE, mind ON DELETE esetén használható.

3. **Bináris 1:1 számosságú kapcsolattípusok leképezése.** Minden bináris 1:1 számosságú *R* kapcsolattípus esetén meg kell határozni az *R*-ben részt vevő egyedtípusokból képzett *S* és *T* relációsémákat. Három lehetséges megközelítés létezik: (a) külső kulcs használata, (b) összevonás és (c) keresztreferencia vagy kapcsoló relációséma használata. Az első megközelítés a leghasznosabb, és azt célszerű alkalmazni, hacsak bizonyos feltételek nem

állnak fenn, ahogy azt mindjárt látni fogjuk:

a. **Külső kulcs használata.** Válasszuk ki az egyik relációsémát (mondjuk *S*-et), és vegyük fel *S* külső kulcsaként *T* elsődleges kulcsát. Célszerű *S*-nek azt a relációsémát választani, amelyiket abból az egyedtípusból képeztünk le, amelyik *totális résztvevője* az *R* kapcsolatnak. Vegyük fel továbbá *R* egyszerű attribútumait, illetve *R* összetett attribútumainak egyszerű komponenseit *S* attribútumaiként.

   Példánkban a 7.1. ábrán látható VEZETI kapcsolattípust úgy képezzük le, hogy *S* helyére az OSZTÁLY egyedtípust választjuk, mert az totális résztvevője a VEZETI kapcsolattípusnak (minden osztálynak van vezetője). Az OSZTÁLY relációsémához hozzávesszük külső kulcsként a DOLGOZÓ relációséma elsődleges kulcsát, és átnevezzük Vez_szsz-re. Szintén hozzávesszük az OSZTÁLY relációsémához a VEZETI kapcsolattípus Kezdő_dátum egyszerű attribútumát is, és átnevezzük Vez_kezdő_dátumra (lásd a 7.2. ábrát).

   Azt is megtehetnénk, hogy *S* (a totális résztvevő) elsődleges kulcsát vesszük fel *T* külső kulcsaként. A példánkban ez azt jelentené, hogy a DOLGOZÓ relációsémában megjelenne mondjuk egy Vezetett_osztály külső kulcs attribútum, amely mindazon dolgozó rekordok esetén NULL értéket tartalmazna, akik nem vezetnek egyetlen osztályt sem. Ha a dolgozóknak csak 10 százaléka vezet osztályt, akkor a külsőkulcs-értékek 90 százaléka NULL érték lenne a mi esetünkben. Harmadik lehetőség lenne, hogy mind *S*-be, mind pedig *T*-be vegyünk be külső kulcsokat redundánsan, de ez plusz konzisztencia-karbantartási terheket róna ránk.

b. **Összevonás.** Egy másik lehetőség az 1:1 kapcsolatok leképezésére, ha a két egyedtípust és a kapcsolatot egyetlen relációba vonjuk össze. Ezt akkor tehetjük meg, ha *mindkét egyedtípus totális résztvevője* a kapcsolatnak.

c. **Kereszthivatkozás vagy kapcsoló relációséma használata .** A harmadik lehetőség, hogy felveszünk egy harmadik *R* relációsémát abból a célból, hogy kereszthivatkozással lássuk el a két egyedtípusból képzett *S* és *T* relációsémák elsődleges kulcsait. Ahogy látni fogjuk, ezt a megközelítést alkalmazzuk a bináris M:N kapcsolatoknál is. Az *R* relációsémát **kapcsoló relációsémának** nevezzük, mert *R* relációpéldányainak minden rekordja egy kapcsolat-előfordulást reprezentál, amely *S* egy relációjának egy rekordját *T* egy relációjának egy rekordjával kapcsolja össze.

4. **Bináris 1:N számosságú kapcsolattípusok leképezése.** Minden bináris 1:N számosságú *R* kapcsolattípus esetén meg kell határozni azt az *S* relációsémát, amelyiket a kapcsolattípus N-oldali egyedtípusából képeztünk. Vegyük fel *S* külső kulcsaként az *R*-ben részt vevő másik egyedtípusból képzett *T* relációséma elsődleges kulcsát; mindezt azért tesszük, mert az N-oldali egyed-előfordulások a másik oldalról legfeljebb egy egyed-előforduláshoz tartoznak. Vegyük fel továbbá *R* egyszerű attribútumait, illetve *R* összetett attribútumainak egyszerű komponenseit *S* attribútumaiként.

   Példánkban most leképezzük a 7.1. ábrán látható MUNKAHELYE, IRÁNYÍTJA és FŐNÖKE 1:N kapcsolattípusokat. A MUNKAHELYE esetén az OSZTÁLY Oszám elsődleges kulcsát hozzávesszük külső kulcsként a DOLGOZÓ relációsémához, és elnevezzük Osz-nek. A FŐNÖKE esetén a DOLGOZÓ relációséma elsődleges kulcsát magához a DOLGOZÓ relációsémához vesszük hozzá külső kulcsként (mert a kapcsolattípus rekurzív), és elnevezzük Főnök_szsz-nek. Az IRÁNYÍTJA kapcsolattípust a PROJEKT Osz nevű külső kulcs attribútumára képezzük le, amely az OSZTÁLY relációséma Oszám elsődleges kulcsára hivatkozik. Ezek a külső kulcsok a 7.2. ábrán láthatók.

   Egy másik megközelítés szerint most is használhatunk kapcsoló relációsémát (kereszthivatkozást), ahogy az 1:1 kapcsolatoknál tettük. Ekkor egy külön *R* relációsémát

hozunk létre, amelynek attribútumai *S* és *T* elsődleges kulcsai, és amelynek elsődleges kulcsa megegyezik *S* elsődleges kulcsával. Ezt a módszert célszerű alkalmazni akkor, ha *S*-ben kevés rekord vesz részt a kapcsolatban, ugyanis ekkor a külső kulcsos megközelítés használatakor rengeteg NULL érték szerepelne a külső kulcsban.

5. **Bináris M:N számosságú kapcsolattípusok leképezése.** Minden bináris M:N számosságú *R* kapcsolattípus esetén hozzunk létre egy új *S* relációsémát, amely *R*-et reprezentálja. Vegyük fel *S* külső kulcsaként a kapcsolatban részt vevő egyedtípusokból képzett relációsémák elsődleges kulcsait; ezek együttese alkotja *S* elsődleges kulcsát. Vegyük fel továbbá *R* egyszerű attribútumait, illetve *R* összetett attribútumainak egyszerű komponenseit *S* attribútumaiként. Vegyük észre, hogy egy M:N kapcsolatot nem tudunk egyetlen külső kulccsal reprezentálni az egyik résztvevő relációsémában (ahogy az 1:1 és 1:N kapcsolattípusok esetén tettük) az M:N számosság miatt; mindenképpen létre kell hoznunk egy külön *S kapcsoló relációsémát*.

   Példánkban a 7.1. ábrán látható DOLGOZIK_RAJTA M:N kapcsolattípust úgy képezzük le, hogy létrehozzuk a 7.2. ábrán látható DOLGOZIK_RAJTA relációsémát. A PROJEKT és a DOLGOZÓ relációsémák elsődleges kulcsait hozzávesszük külső kulcsként a DOLGOZIK_RAJTA relációsémához, és átnevezzük őket Psz-re, illetve Dszsz-re. Ezenkívül felveszünk még egy Órák attribútumot is a DOLHOZIK_RAJTA relációsémába, amely a kapcsolattípus Órák attribútumát reprezentálja. A DOLGOZIK_RAJTA relációséma elsődleges kulcsa a külső kulcs attribútumok {Dszsz, Psz} együttese lesz. Ezt a *kapcsoló* relácsiósémát a 7.3. (c) ábrán láthatjuk.

   A CASCADE opciót célszerű megadni a hivatkozó táblában bekövetkező eseményre az *R* kapcsolattípusnak megfelelő relációséma külső kulcsának definiálásakor, mivel minden kapcsolat-előfordulás létezésfüggően kapcsolódik a benne szereplő egyed-előfordulásokhoz. Ez az opció mind ON UPDATE, mind ON DELETE esetén használható.

   Azt is vegyük észre, hogy az 1:1 és 1:N kapcsolattípusokat mindig leképezhetjük az M:N kapcsolattípusok leképezéséhez hasonló módon, a kereszthivatkozás (kapcsoló relációséma) használatával úgy, ahogy azt korábban láttuk. Ez a megközelítés különösen hasznos akkor, ha kevés kapcsolat-előfordulás létezik, mert így elkerülhetjük a sok NULL érték felbukkanását a külső kulcsokban. Ebben az esetben a kapcsoló relációséma elsődleges kulcsát a kapcsolatban részt vevő egyedtípusokból képzett relációkra hivatkozó külső kulcsok közül *csak az egyik* alkotja. 1:N számosságú kapcsolattípus esetén a kapcsoló relációséma elsődleges kulcsa az N-oldali egyedtípusból képzett relációsémára hivatkozó külső kulcs lesz. 1:1 számosságú kapcsolattípus esetén bármelyik külső kulcsot használhatjuk a kapcsoló relációséma elsődleges kulcsaként, ha a hozzá tartozó relációban nincs NULL érték.

6. **Többértékű attribútumok leképezése.** Minden egyes *A* többértékű attribútum esetén hozzunk létre egy új *R* relációsémát. Ez az *R* relációséma tartalmazzon egy, az *A*-nak megfelelő attribútumot, valamint annak a relációsémának a *K* elsődleges kulcsát — *R* külső kulcsaként —, amelyet az *A*-t tartalmazó egyedtípusból vagy kapcsolattípusból képeztünk. *R* elsődleges kulcsát *A* és *K* együttese alkotja. Ha a többértékű attribútum összetett, akkor az egyszerű komponenseit vegyük fel *R* attribútumaiként.

   A példánkban létrehozunk egy OSZT_HELYSZÍNEK relációsémát (lásd a 7.3. (d) ábrát). Az Ohelyszín attribútum reprezentálja az OSZTÁLY többértékű Helyszínek attribútumát, az Oszám attribútum pedug az OSZTÁLY relációséma elsődleges kulcsát (külső kulcsként). Az OSZT_HELYSZÍNEK elsődleges kulcsa az {Oszám, Ohelyszín} kombináció lesz. Az OSZT_HELYSZÍNEK relációiban egy osztály minden egyes helyszínéhez külön rekord fog tartozni.

   A CASCADE opciót célszerű megadni a hivatkozó táblában bekövetkező eseményre a

többértékű attribútumnak megfelelő relációséma külső kulcsának definiálásakor mind ON UPDATE, mind ON DELETE esetén. Meg kell jegyeznünk, hogy egy összetett, többértékű attribútum leképezése esetén *R* kulcsának meghatározása a komponens attribútumok jelentésének az elemzését követeli meg. Bizonyos esetekben, ha egy többértékű attribútum összetett is, csak néhány komponens attribútumot kell közülük *R* kulcsához hozzávenni; ezek az attribútumok hasonlítanak egy olyan gyenge egyedtípus részleges kulcsához, amelyek a többértékű attribútumnak felelnek meg (lásd a „Gyenge egyedtípusok” alfejezetet).

A 7.2. ábrán az 1-től 6-ig leírt lépések alkalmazása után kapott VÁLLALAT relációs adatbáziséma, a 5.4. ábrán pedig egy példa adatbázisállapot látható. Szemfüles olvasóink észrevehették, hogy még nem beszéltünk az *n*-edfokú kapcsolattípusok leképezéséről ($n > 2$), mivel ilyen nincsen a 7.1. ábrán; ezeket az M:N kapcsolattípusokhoz hasonló módon kell leképezni, a következő lépést is hozzávéve a leképezési algoritmusunkhoz.

7. **Az *n*-edfokú kapcsolattípusok leképezése.** Minden *n*-edfokú *R* kapcsolattípus esetén, ahol $n > 2$, hozzunk létre egy új *S* relációsémát, amely *R*-et reprezentálja. Vegyük fel *S* külső kulcsaként a kapcsolatban részt vevő egyedtípusokból képzett relációsémák elsődleges kulcsait. Vegyük fel továbbá *R* egyszerű attribútumait, illetve *R* összetett attribútumainak egyszerű komponenseit *S* attribútumaiként. *S* elsődleges kulcsa általában az összes külső kulcs együttese. Ha azonban az *R*-ben részt vevő valamely *E* egyedtípusból csak egy rekord vesz részt a kapcsolatban (számossági megszorítás), akkor *S* elsődleges kulcsának nem kell tartalmaznia az *E*-ből képzett *E*' relációsémára hivatkozó külső kulcsot (lásd a . alfejezetet).

   Tekintsük például a 3.16. ábrán látható SZÁLLÍTÁS kapcsolattípust. Ezt a 7.4. ábrán látható SZÁLLÍTÁS relációsémára képezhetjük le, amelynek az elsődleges kulcsa a három külső kulcs együttese lesz: {Bnév, Alkatrész_azon, Proj_név}.

7.4. ábra - A 3.16. (a) ábra *n*-edfokú SZÁLLÍT kapcsolattípusának leképezése.

Összegzés az ER modell konstrukcióinak a leképezéséhez

A 7.1. táblázat összefoglalja az ER és a relációs modellbeli konstrukciók és megszorítások közötti megfeleltetéseket.

7.1. táblázat - Megfeleltetések az ER és a relációs modell között

ER modell	Relációs modell
Egyedtípus	*Egyedtípusból képzett* relációséma
1:1 vagy 1:N kapcsolattípus	Külső kulcs (vagy *kapcsoló* relációséma)

ER modell	**Relációs modell**
M:N kapcsolattípus	*Kapcsoló* relációséma és *két* külső kulcs
n-edfokú kapcsolattípus	*Kapcsoló* relációséma és *n* külső kulcs
Egyszerű attribútum	Attribútum
Összetett attribútum	Egyszerű komponens attribútumok
Többértékű attribútum	Relációséma és külső kulcs
Értékhalmaz	Tartomány
Kulcs attribútum	Elsődleges (vagy másodlagos) kulcs

Az egyik legfőbb megjegyzés az, hogy a relációs sémában, az ER sémával szemben, a kapcsolattípusokat nem reprezentáljuk explicit módon; ehelyett ezeket a két relációsémában (*S*-ben és *T*-ben) szereplő *A* elsődleges kulcs és az ugyanazon tartományon értelmezett *B* külső kulcs attribútumhalmazokkal reprezentáljuk. Az *S*-hez és *T*-hez tartozó relációk egy-egy rekordja akkor kapcsolódik egymáshoz, ha ugyanazon értékekkel rendelkeznek *A*-n és *B*-n. Az *S*-hez és *T*-hez tartozó relációkat az *S.A* és *T.B* attribútumok alapján egy equijoinnal összekapcsolva (vagy természetes összekapcsolással, ha a két kapcsoló attribútumnak ugyanaz a neve) egymáshoz rendelhetjük az összekapcsolt relációk összetartozó rekordpárjait, és így materializálhatjuk a kapcsolatot. Ha bináris 1:1 vagy 1:N kapcsolattípusról van szó, akkor általában csak egy összekapcsolási művelet szükséges. Egy bináris M:N kapcsolattípus esetén viszont két összekapcsolásra, míg az *n*-edfokú kapcsolattípusok esetén *n* összekapcsolásra van szükségünk ahhoz, hogy teljesen materializálhassuk a kapcsolat-előfordulásokat.

Ahhoz például, hogy képezzünk egy olyan relációt, amely tartalmazza a dolgozó nevét, a projekt nevét, és hogy az egyes dolgozók hány órát dolgoznak az egyes projekteken, össze kell kötnünk a DOLGOZÓ rekordjait a kapcsolódó PROJEKT rekordokkal a DOLGOZIK_RAJTA reláción keresztül (lásd a 7.2. ábrát). Az equijoin műveletet tehát a DOLGOZÓ és a DOLGOZIK_RAJTA relációkra kell végrehajtani az Szsz = Dszsz összekapcsolási feltétellel, majd az eredményül kapott relációra és a PROJEKT relációra egy újabb equijoin műveletet kell alkalmazni a Psz = Pszám összekapcsolási feltétellel. Minél több kapcsolatot szeretnénk érinteni, általában annál több összekapcsolási műveletet kell megadnunk. A relációs adatbázis felhasználójának mindig nagyon ügyelnie kell arra, hogy a külső kulcs attribútumokat helyesen használja kettő vagy több reláció kapcsolódó rekordjainak összerendelésekor. Ezt néha a relációs adatmodell hátrányának tekintik, mert a külső kulcs/elsődleges kulcs megfeleltetések nem mindig nyilvánvalóak a relációs sémák vizsgálatakor. Ha végrehajtunk egy equijoint két reláció olyan attribútumai alapján, amelyek nem alkotnak külső kulcs/elsődleges kulcs kapcsolatot, az eredmény gyakran értelmetlen lesz, és hamis adatokat eredményezhet. Az olvasó például megpróbálhatja összekapcsolni a PROJEKT és az OSZT_HELYSZÍNEK relációkat az Ohelyszín = Phelyszín feltétel alapján, és megvizsgálni az eredményt (lásd a 8. fejezetet).

A relációs sémában *minden egyes* többértékű attribútumhoz külön relációsémát hozunk létre. Egy konkrét egyed esetén, ahol a többértékű attribútum értéke egy értékhalmaz, az egyed kulcs attribútumának értékét külön rekordokban kell megismételnünk az értékhalmaz minden egyes elemével, mivel az alap relációs modell *nem* enged meg egy rekordban egy attribútumhoz több értéket (értékek egy listáját vagy halmazát). Például mivel az 5-ös osztálynak három helyszíne van, három rá vonatkozó rekord létezik az 5.4. ábra OSZT_HELYSZÍNEK relációjában; minden rekord egy helyszínt ad meg. A példánkban egy equijoint hajthatunk végre az OSZT_HELYSZÍNEK és az OSZTÁLY relációkra az Oszám attribútum alapján, hogy megkapjuk az összes helyszínt más OSZTÁLY-beli attribútumokkal együtt. Az eredményül kapott relációban az OSZTÁLY egyéb attribútumainak az értékei külön rekordokban ismétlődnek az egyes osztályok minden egyes helyszínére.

Az alap relációalgebra nem rendelkezik a beágyazás vagy tömörítés művelettel, amely a { ⟨1,

'Budapest'⟩, ⟨4, 'Kecskemét'⟩, ⟨5, {'Vác', 'Tiszafüred', 'Budapest'}⟩ } formájú rekordhalmazt eredményezné az 5.4. ábra OSZT_HELYSZÍNEK relációjából. Ez egy komoly hátránya a relációs modell normalizált vagy *lapos* alapváltozatának. A beágyazás kifejezésére az objektumorientált modellek és a korai hierarchikus és hálós modellek jobb eszközrendszerrel rendelkeznek, mint a relációs modell. A beágyazott relációs modell és az objektum-relációs rendszerek megpróbálják orvosolni ezt a problémát.

Az EER modellbeli konstrukciók leképezése relációsémákra

A következőkben az EER modellbeli konstrukciók relációs sémákra történő leképezését tárgyaljuk a . alfejezetben ismertetett ER-ről relációsra történő leképezési algoritmus kibővítésével.

A specializációk és generalizációk leképezése

Számos módszer létezik az olyan alosztályok leképezésére, amelyek együtt egy specializációt alkotnak (vagy másképpen, amelyek generalizálva vannak egy szuperosztályba), mint például a 4.4. ábrán látható DOLGOZÓ-nak a {TITKÁR(NŐ), TECHNIKUS, MÉRNÖK} alosztályai. Hozzáadhatunk egy további lépést a . alfejezetben leírt ER-ről relációsra történő leképezési algoritmusunkhoz, amely a specializációk leképezését kezeli. A 8. lépés a legelterjedtebb opciókat mutatja be; más leképezések is lehetségesek. Megadjuk azokat a feltételeket, amelyek teljesülése esetén az egyes opciók használhatók. Az Attrs(R) jelölést az *R relációséma attribútumainak*, PK(R)-et pedig az *R elsődleges kulcsának* a jelölésére fogjuk használni. Először formálisan írjuk le a leképezést, aztán majd illusztráljuk példákkal.

8. **Lehetőségek specializációk és generalizációk leképezésére.** Konvertáljunk át relációsémákká minden C (generalizált) szuperosztállyal és m darab, $\{S_1, S_2, \ldots, S_m\}$ alosztállyal rendelkező specializációt, ahol C attribútumai $\{k, a_1, \ldots, a_n\}$ és k az (elsődleges) kulcs, a következő lehetőségek szerint:
   - **8A opció: Több relációséma — szuperosztály és alosztályok.** Hozzunk létre egy L relációsémát a C számára Attrs(L) = $\{k, a_1, \ldots, a_n\}$ attribútumokkal és PK(L) = k elsődleges kulccsal. Hozzunk létre egy L_i relációsémát minden egyes S_i alosztályhoz ($1 \le i \le m$) Attrs(L_i) = $\{k\} \cup \{S_i$ attribútumai$\}$ attribútumokkal és PK(L_i) = k elsődleges kulccsal. Ez a lehetőség mindeféle specializáció esetén (totális vagy részleges, kizáró vagy átfedő) működik.
   - **8B opció: Több relációséma — csak alosztály relációsémák.** Hozzunk létre egy L_i relációsémát minden egyes S_i alosztályhoz ($1 \le i \le m$) Attrs(L_i) = $\{S_i$ attribútumai$\} \cup \{k, a_1, \ldots, a_n\}$ attribútumokkal és PK(L_i) = k elsődleges kulccsal. Ez a lehetőség csak olyan specializáció esetén működik, ahol az alosztályok *totálisak* (minden szuperosztálybeli egyednek legalább egy alosztályhoz kell tartoznia). Ha a specializáció *átfedő*, egy egyed több relációban is felbukkanhat.
   - **8C opció: egyetlen relációséma egy típus attribútummal.** Hozzunk létre egy L relációsémát Attrs(L) = $\{k, a_1, \ldots, a_n\} \cup \{S_1$ attribútumai$\} \cup \ldots \cup \{S_n$ attribútumai$\} \cup \{t\}$ attribútumokkal és PK(L) = k elsődleges kulccsal. A t-t **típus** (vagy **diszkrimináló**) attribútumnak nevezzük, amely jelzi azt az alosztályt, amelyhez az egyes rekordok tartoznak, ha van ilyen egyáltalán. Ez a lehetőség csak olyan specializáció esetén működik, amely *kizáró*, és fennáll a veszélye annak, hogy sok NULL értéket generál, ha sok saját attribútum szerepel az alosztályban.
   - **8D opció: egyetlen relációséma több típus attribútummal.** Hozzunk létre egy L

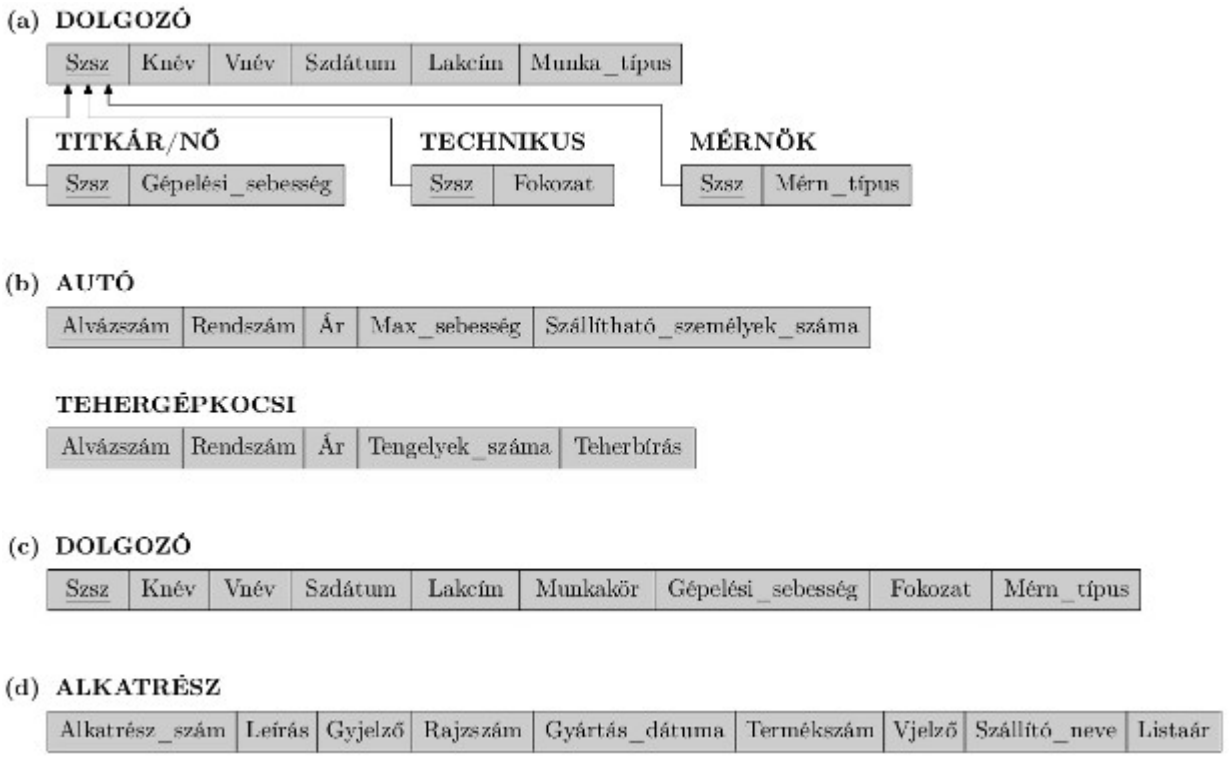
relációsémát Attrs(L) = $\{k, a_1, \ldots, a_n\} \cup \{S_1$ attribútumai$\} \cup \ldots \cup \{S_n$ attribútumai$\} \cup \{t_1, t_2, \ldots t_m\}$ attribútumokkal és PK(L) = k elsődleges kulccsal. Minden t_i ($1 \leq i \leq m$) **logikai típusú típus attribútum**, amely azt jelzi, hogy egy adott rekord az S_i alosztályhoz tartozik-e. Ez a lehetőség olyan specializációk esetén is működik, amely *átfedő* alosztályokat tartalmaz (de nyilván működik kizáró specializációk esetén is).

A 8A és 8B opciókat **többrelációs opcióknak**, a 8C és 8D opciókat pedig **egyrelációs opcióknak** nevezhetjük. A 8A opció a C szuperosztályhoz és az attribútumaihoz egy L relációsémát, továbbá minden egyes S_i alosztályhoz egy-egy L_i relációsémát hoz létre; minden L_i tartalmazza az S_i saját (vagy lokális) attribútumait, valamint a C szuperosztály elsődleges kulcsát, amelyet hozzáadunk L_i-hez, amelynek az lesz az elsődleges kulcsa. Egyúttal a szuperosztály relációsémájára hivatkozó külső kulcs is lesz. Bármelyik L_i és L között, az elsődleges kulcs alapján végrehajtott equijoin művelet előállítja az S_i-beli egyedek összes saját és örökölt attribűtumát. Ezt az opciót a 7.5. (a) ábra illusztrálja a 4.4. ábrán látható EER séma leképezésével. A 8A opció a specializáció bármilyen megszorításával működik: kizáró vagy átfedő, totális vagy részleges. Vegyük észre, hogy a

$$\pi_{\langle k \rangle}(L_i) \subseteq \pi_{\langle k \rangle}(L)$$

megszorítás minden egyes L_i-re fennáll. Ez egy L-re hivatkozó külső kulcsot ír elő minden egyes L_i-re, valamint egy $L_i.K < L.K$ *tartalmazásfüggést* is definiál (lásd a . alfejezetet).

7.5. ábra - A specializációk és generalizációk leképezésének lehetőségei. (a) A 4.4. ábrán látható EER séma leképezése a 8A opcióval. (b) A 4.3. (b) ábrán látható EER séma leképezése a 8B opcióval. (c) A 4.4. ábrán látható EER séma leképezése a 8C opcióval. (d) A 4.5. ábrán látható EER séma leképezése a logikai típusú Gyjelző és Vjelző típus mezőkkel.



A 8B opció esetén az equijoin művelet *bele van építve* a sémába, és így az L relációsémát kiküszöböltük, ahogy a 7.5. (b) ábrán látható, amely a 4.3. (b) ábra EER specializációjának a leképezését mutatja. Ez az opció csak akkor működik jól, ha a specializáció *egyszerre* kizáró és

totális. Ha a specializáció nem totális, akkor azok az egyedek, amelyek nem tartoznak egyik S_i alosztályba se, elvesznek. Ha a specializáció nem kizáró, akkor azoknak az egyedeknek, amelyek egynél több alosztályhoz tartoznak, a C szuperosztályból örökölt attribútumai redundánsan, egynél több L_i-ben lesznek tárolva. A 8B opció esetén egyetlen reláció sem tartalmazza a C szuperosztály összes egyedét; emiatt egy külső összekapcsolás (vagy teljes külső összekapcsolás) műveletet kell alkalmaznunk az L_i relációira, hogy megkapjuk a C szuperosztály összes egyedét. A külső összekapcsolás eredménye hasonló lesz a 8C és 8D opciókkal kapott relációkhoz, kivéve hogy nem lesznek típus mezők. Ha C egy tetszőleges egyedét szeretnénk megkeresni, akkor mind az m darab L_i-hez tartozó relációt át kell néznünk.

A 8C és 8D opciók egyetlen relációsémát hoznak létre a C szuperosztálynak és az összes alosztályának reprezentálására. Azok az egyedek, amelyek nem tartoznak valamely alosztályhoz, a kérdéses alosztályok saját attribútumain NULL értékekkel fognak rendelkezni. Ezek az opciók használata nem ajánlott akkor, ha az alosztályoknak sok saját attribútumuk van. Ha azonban kevés saját attribútum létezik, ezek a leképezések előnyösebbek a 8A és 8B opciókhoz képest, mert nem szükséges equijoin és külső összekapcsolás műveleteket megadni, és emiatt hatékonyabb implementációt érhetünk el.

A 8C opció kizáró alosztályok kezelésére használható egy t **típus** (vagy **diszkrimináló**) **attribútum** bevezetésével, amely azt jelzi, hogy az egyes rekordok melyik alosztályhoz tartoznak; t tartománya így az {1, 2, …, m} halmaz lehet. Ha a specializáció részleges, t NULL értékkel fog rendelkezni az olyan rekordokban, amelyek egyetlen alosztályhoz sem tartoznak. Ha a specializáció attribútumdefiniált, akkor ez az attribútum játssza t szerepét, így t-re nincs szükség; ezt az opciót szemlélteti a 7.5. (c) ábra, amelyen a 4.4. ábra EER specializációjának leképezése látható.

A 8D opciót arra tervezték, hogy átfedő alosztályokat kezeljen m darab logikai típusú típus mező bevezetésével, *minden egyes* alosztályhoz eggyel. Használható kizáró alosztályok esetén is. A t_i típus mezők az {igen, nem} tartománnyal rendelkezhetnek, ahol az igen érték jelzi, hogy az adott rekord az S_i alosztály tagja. Ha a 4.4. ábra EER specializációjára ezt az opciót alkalmaznánk, három típus attribútumot vezetnénk be (Titkár(nő)_e, Mérnök_e, Technikus_e) a 7.5. (c) ábrán látható Munkakör attribútum helyett. Megjegyezzük, hogy az m darab típus mező helyett elegendő egyetlen, m *bitből* álló típus attribútumot is létrehozni.

7.6. ábra - A 4.7. ábra EER specializációs hálójának leképezése többféle opció használatával.

Ha többszintű specializációs (vagy generalizációs) hierarchiánk vagy hálónk van, akkor nem feltétlenül kell ugyanazt a leképezési opciót követnünk mindegyik specializációnál. Ehelyett a hierarchia vagy háló egyik részére az egyik leképezési opciót használhatjuk, míg más részeire másokat. A 7.6. ábra a 4.7. ábrán látható EER háló relációsémákra történő egy lehetséges leképezését mutatja. Itt a 8A opciót használtuk a SZEMÉLY/{DOLGOZÓ, ÖREGDIÁK,

HALLGATÓ}, a 8C opciót a DOLGOZÓ/{IRODAI_DOLGOZÓ, OKTATÓ, DEMONSTRÁTOR}, míg a 8D opciót a DEMONSTRÁTOR/{SEGÉDKUTATÓ, SEGÉDOKTATÓ}, a HALLGATÓ/DEMONSTRÁTOR (a HALLGATÓ relációsémában) és a HALLGATÓ/{VÉGZŐS_HALLGATÓ, NEM_VÉGZŐS_HALLGATÓ} specializációk leképezéséhez. A 7.6. ábrán minden olyan attribútum, amelynek a neve *típusra* vagy *jelzőre* végződik, típus mező.

Osztott alosztályok (többszörös öröklődés) leképezése

Az osztott alosztály, mint amilyen a 4.7. ábrán látható MÉRNÖK_IGAZGATÓ, több szuperosztály alosztálya, amely többszörös öröklődést jelez. Ezen osztályok mindegyikének ugyanazzal a kulcs attribútummal kell rendelkeznie; máskülönben az osztott alosztályt kategóriaként modelleznénk. Az osztott alosztályokra a 8. lépésben ismertetett bármelyik opciót alkalmazhatjuk, figyelembe véve a leképezési algoritmus 8. lépésében tárgyalt megkötéseket. A 7.6. ábrán a 8C és 8D opciókat használtuk a DEMONSTRÁTOR osztott alosztály leképezéséhez. A 8C opciót a DOLGOZÓ relációsémában (Dolgozó_típus attribútum), a 8D opciót pedig a HALLGATÓ relációsémában (Demonstrátor_jelző attribútum) használtuk.

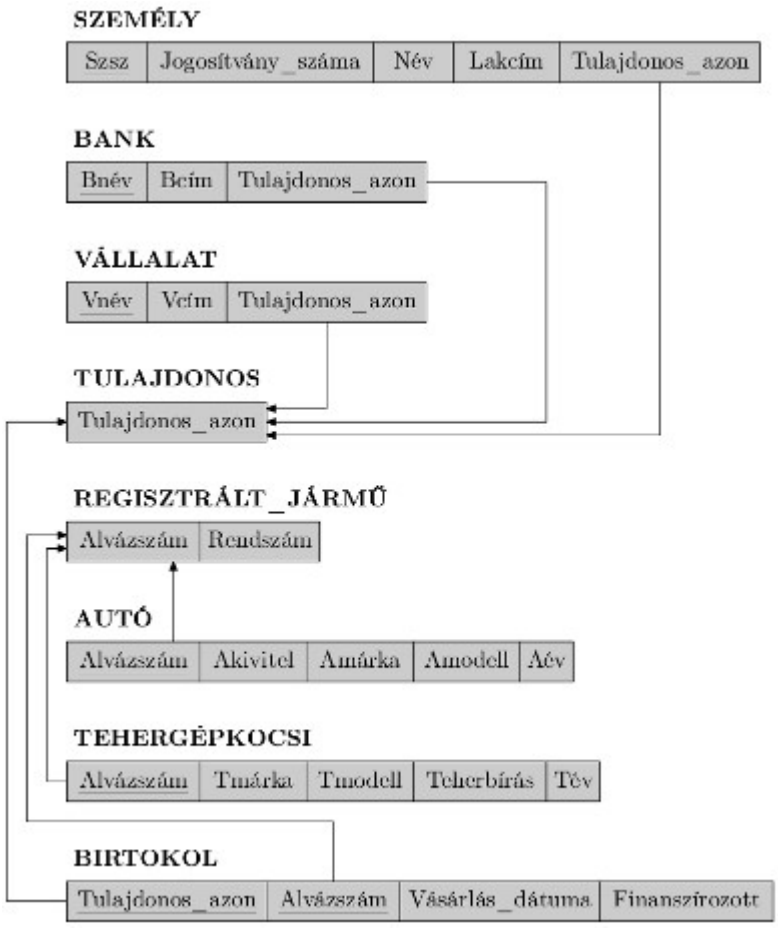
Kategóriák (unió típusok) leképezése

A kategóriák kezelésére egy további lépést (9. lépés) adunk a leképezési eljárásunkhoz. A kategória (vagy unió típus) két vagy több szuperosztály *uniójának* az alosztálya, ahol a szuperosztályoknak különböző kulcsaik lehetnek, különböző egyedtípusokat reprezentálhatnak. Példa rá a 4.8. ábrán látható TULAJDONOS kategória, amely a SZEMÉLY, a BANK és a VÁLLALAT egyedtípusok uniójának részhalmaza. Ugyanezen az ábrán a másik kategória, a REGISZTRÁLT_JÁRMŰ két szuperosztállyal rendelkezik, amelyeknek ugyanaz a kulcs attribútumuk.

9. **Az unió típusok (kategóriák) leképezése.** Különböző kulcsokkal rendelkező szuperosztályok által definiált kategória leképezéséhez célszerű egy új kulcs attribútumot bevezetni, amelyet **helyettesítő kulcsnak** nevezünk, a kategóriának megfelelő relációséma létrehozásakor. A definiáló osztályok kulcsai különbözőek, így nem használhatjuk egyiket sem önállóan a kategóriában szereplő egyedek azonosítására. A 4.8. ábrán látható példánkban létrehozhatunk egy TULAJDONOS relációsémát, amely a TULAJDONOS kategóriának felel meg, ahogy a 7.7. ábrán látható, és felvehetjük ebbe a relációsémába a kategória összes attribútumát. A TULAJDONOS relációséma elsődleges kulcsa a helyettesítő kulcs lesz, amelyet Tulajdonos_azonnak hívunk. A Tulajdonos_azon helyettesítő kulcs attribútumot a kategória szuperosztályainak megfelelő relációsémákba is felvesszük külső kulcsként azért, hogy ezzel megadjuk a megfeleltetést a helyettesítő kulcs értékei és az egyes szuperosztályok kulcsértékei között. Vegyük észre, hogy ha egy konkrét SZEMÉLY (vagy BANK, vagy VÁLLALAT) egyed nem tagja a TULAJDONOS kategóriának, akkor a SZEMÉLY (vagy BANK, vagy VÁLLALAT) relációsémához tartozó relációban a neki megfelelő rekord Tulajdonos_azon attribútuma NULL értéket vesz fel, és nem lesz rekordja a TULAJDONOS relációsémához tartozó relációban.

Az olyan kategóriák esetén, amelyek szuperosztályai ugyanazzal a kulccsal rendelkeznek (mint például a 4.8. ábrán a REGISZTRÁLT_JÁRMŰ), nincs szükség helyettesítő kulcsra. A REGISZTRÁLT_JÁRMŰ kategória leképezését szintén a 7.7. ábra mutatja, amely ezt az esetet illusztrálja.

7.7. ábra - A 4.8. ábrán látható EER kategóriák (unió típusok) leképezése relációsémákra.



Összefoglalás

A . alfejezetben megmutattuk, hogyan lehet leképezni egy ER modellbeli koncepcionális sémát relációs adatbázissémára. Megadtunk egy algoritmust az ER-ről relációsra történő leképezéshez, és illusztráltuk azt a VÁLLALAT adatbázisból vett példákkal. A 7.1. táblázat összefoglalta az ER és a relációs modellbeli konstrukciók és megszorítások közötti megfeleltetéseket. Ezután további lépésekkel egészítettük ki az algoritmust a alfejezetben az EER modellbeli konstrukciók relációs modellre történő leképezéséhez. Hasonló algoritmusokat használnak a grafikus adatbázis-tervező eszközök, hogy egy koncepcionális sématervből automatikusan állítsanak elő egy relációs sémát.

Áttekintő kérdések

1. Fejtse ki az ER modellbeli és a relációs modellbeli konstrukciók közötti összefüggéseket! Mutassa meg, hogyan képezhetők le az egyes ER modellbeli szerkezetek a relációs modellre, és adja meg az alternatív leképezési lehetőségeket!
2. Ismertesse az EER modellbeli konstrukciók leképezési lehetőségeit a relációs modellre!

Feladatok

1. Próbálja meg leképezni a 6.7. ábrán látható relációs sémát leképezni ER sémára! Ez a „*reverse engineering*” nevű folyamat része, amikor egy létező adatbázis-implementációból készítünk koncepcionális sémát. Írja fel a menet közben megfogalmazódó feltételezéseit!
2. A 7.8. ábrán egy olyan adatbázis ER sémája látható, amelyet tengerészeti hatóságok használhatnak szállítóhajók nyilvántartására és a helyzetük nyomkövetésére. Képezze le ezt

a sémát relációs sémára, és adja meg az összes elsődleges és külső kulcsot!

7.8. ábra - ER séma a HAJÓKÖZLEKEDÉS adatbázishoz.

3. Képezze le a 3.8 feladat BANK ER sémáját (ami a 3.20. ábrán látható) relációs sémára! Adja meg az összes elsődleges és külső kulcsot! Hajtsa végre ugyanezt a 3.4 feladat REPÜLŐGÉP-MENETREND sémájára (3.19. ábra), valamint a 3.1-tól 3.9-ig terjedő feladatok sémáira is!
4. Képezze le a 4.9. és 4.10. ábrákon látható EER diagramokat relációs sémákra! Magyarázza meg, hogy mi alapján döntött az egyes leképezési opciók mellett!
5. Le lehet-e képezni egy bináris M:N kapcsolattípust új relációséma létrehozása nélkül? Miért vagy miért nem?
6. A 4. fejezet 4.5 feladatában látható ER diagramhoz megadott attribútumok felhasználásával képezze le a teljes sémát relációsémákra! Válasszon egy megfelelő opciót a generalizációk leképezéséhez a . alfejezetben leírt 8A–8D opciók közül, és indokolja választását!

8. fejezet - Funkcionális függések és normalizálás

Tartalom

Az 5. fejezettől a 7. fejezetig a relációs modell és a hozzá kötődő nyelvek különböző aspektusait tárgyaltuk. Minden *relációséma* attribútumokból, a *relációs adatbázisséma* pedig relációsémákból

áll. Eddig azt feltételeztük, hogy az attribútumok az adatbázis-tervező elgondolása vagy pedig egy koncepcionális adatmodellben (például ER-ben vagy EER-ben) felírt adatbázisséma-terv leképezése alapján lettek csoportosítva egy relációsémába. Ezekben a modellekben a tervező azonosítja be az egyedtípusokat és a kapcsolattípusokat, valamint ezek attribútumait, és ez az attribútumok természetes és logikus csoportosítását eredményezi a relációkban, amikor a 7. fejezetben ismertetett leképezési eljárásokat követjük. Ennek ellenére szükségünk van egy olyan formális mértékre, amely megmondja, hogy miért lehet jobb az attribútumok relációsémákba történő egyik csoportosítása, mint a másik. Eddig a koncepcionális tervekről (3. és 4. fejezet) és azok relációs modellre történő leképezéséről (7. fejezet) szóló tárgyalásainkban nem adtunk a megfelelőségre vagy *jóságra* semmilyen mérőszámot, amellyel a tervünk minőségét mérhetnénk, csak a tervező intuícióját. Ebben a fejezetben olyan elméleti ismeretekről tárgyalunk, amelyeknek célja a relációsémák kiértékelése a terv minősége szempontjából — azaz hogy formálisan megmérjük, hogy miért jobb az attribútumok relációsémákba történő egyik csoportosítása, mint a másik.

Két szinten beszélhetünk a relációsémák *jóságáról*. Az egyik a **logikai** (vagy **koncepcionális**) **szint**, amely azt írja le, hogy a felhasználók hogyan értelmezik a relációsémákat és az attribútumaik jelentését. Ha ezen a szinten jó relációsémáink vannak, akkor lehetővé válik, hogy a felhasználók tisztában legyenek a relációkban szereplő adatok jelentésével, és ezáltal helyesen tudják formalizálni a lekérdezéseiket. A másik az **implementációs** (vagy **tárolási**) **szint**, amely azt írja le, hogy hogyan tárolódnak és módosulnak a rekordok az alaprelációkban. Ez a szint csak a (fizikailag, állományokban tárolt) alaprelációk sémáira vonatkozik, míg a logikai szinten mind az alaprelációk, mind a nézetek (virtuális relációk) sémáira kíváncsiak vagyunk. Az ebben a fejezetben ismertetett relációs adatbázis-tervezési elmélet elsősorban az *alaprelációkra* vonatkozik, bár a megfelelőség bizonyos kritériumai a nézetekre is érvényesek, ahogyan arról majd a . alfejezetben írunk.

Sok más tervezési feladathoz hasonlóan az adatbázis-tervezés is két megközelítést felhasználva hajtható végre: alulról felfelé vagy felülről lefelé. Az **alulról felfelé tervezési módszer** (amelyet *szintézis alapú tervezésnek* is neveznek) *az egyes attribútumok közötti* alapvető kapcsolatokat tekinti kiindulópontnak, és azokat használja fel a relációsémák építéséhez. A gyakorlatban ez a megközelítés nem igazán népszerű[26], mert az a probléma vele, hogy kiindulásképpen össze kell gyűjteni egy csomó bináris kapcsolatot az attribútumok között. Ezzel szemben a **felülről lefelé tervezési módszer** (amelyet *analízis alapú tervezésnek* is neveznek) számos, relációkba szervezett attribútumcsoportból indul ki, amelyek természetes módon összetartoznak, mint például egy számla, egy űrlap vagy egy jelentés. A relációkat ezután külön-külön és együtt is elemezzük, amely további dekompozíciókhoz vezet mindaddig, amíg el nem érjük az összes kívánt tulajdonságot. Az ebben a fejezetben tárgyalt elmélet mind a felülről lefelé, mind az alulról felfelé tervezési megközelítés esetében alkalmazható, de praktikusabb felülről lefelé való tervezéssel használni.

A fejezetet azzal kezdjük, hogy informálisan bemutatjuk a jó és a rossz relációsémák néhány ismertetőjegyét a . alfejezetben. A . alfejezetben definiáljuk a *funkcionális függés* fogalmát, amely egy attribútumok közötti formális megszorítás; ez a megszorítás a legfőbb eszköz arra, hogy formálisan megmérjük az attribútumok relációsémákba történő csoportosításának a megfelelőségét. A funkcionális függések tulajdonságait is tanulmányozzuk és elemezzük. A . alfejezetben a normálformákat és a funkcionális függéseken alapuló normalizálási folyamatot ismertetjük. Ezután további normálformákat definiálunk, hogy a relációsémák megfeleljenek bizonyos elvárt megszorításoknak, amelyeket funkcionális függésekkel fejezünk ki. A normalizálási eljárás a relációkra vonatkozó tesztek sorozatának az alkalmazásából áll, amelyek ellenőrzik, hogy a relációk megfelelnek-e ezeknek az egyre szigorodó követelményeknek, és ha szükséges, felbontják a relációkat. A . alfejezetben a normálformák általánosabb definícióit ismertetjük, amelyek közvetlenül alkalmazhatók bármely tervre, és nem igénylik a lépésről lépésre végrehajtott elemzést és normalizálást.

A 9. fejezetben folytatjuk a jó relációsémák tervezéséhez kapcsolódó elmélet tárgyalását. Ismertetjük a relációk dekompozíciójának elvárt tulajdonságait (nemadditív join tulajdonság és a

funkcionális függések megőrzésének tulajdonsága), majd néhány algoritmus bemutatásával rátérünk az adatbázis-tervezés alulról felfelé történő megközelítésére. Ezek az algoritmusok bemenetként funkcionális függések egy halmazát kapják, és egy kívánt normálformájú relációs tervet eredményeznek, miközben betartják a fent említett elvárt tulajdonságokat. Bemutatunk egy általános algoritmust annak eldöntésére, hogy egy dekompozíció rendelkezik-e a veszteségmentes join tulajdonsággal. A 9. fejezetben további függéstípusokat és magasabb normálformákat is definiálunk, amelyek tovább javítják a relációsémák *jóságát*.

Azon olvasók, akik csak a normalizálás informális bevezetése iránt érdeklődnek, a . és . alfejezeteket átugorhatják. Ha a kurzus nem fedi le a 9. fejezetet, javasoljuk a 8. fejezeten kívül a dekompozíció elvárt tulajdonságainak gyors áttekintését a . alfejezetben, valamint a nemadditív join tulajdonság bináris dekompozíciók esetén történő tesztelésének az áttanulmányozását.

Nem hivatalos tervezési irányelvek relációsémákhoz

Mielőtt belekezdenénk a relációs adatbázis-tervezés formális elméletének tárgyalásába, ebben az alfejezetben négy, a relációs sématervezés minőségére vonatkozó *informális mérőszámot* mutatunk be:

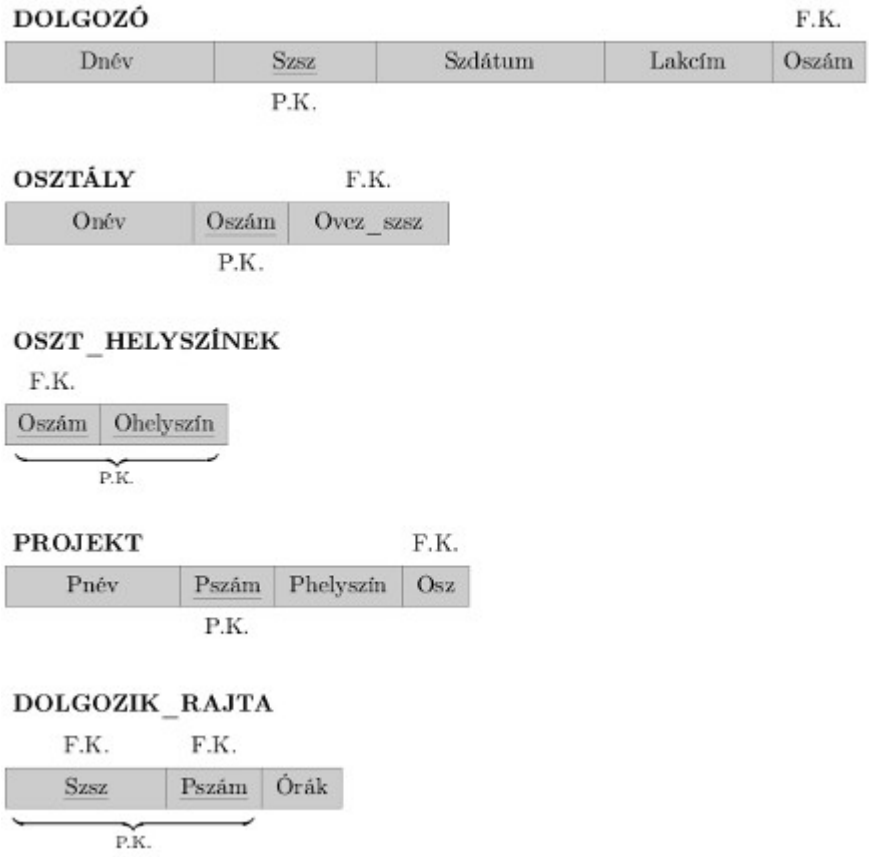
- az attribútumok szemantikájának az érthetősége;
- a rekordokban megjelenő redundáns információk mennyisége;
- a rekordokban megjelenő NULL értékek mennyisége;
- az álrekord-generálási lehetőség kizárása.

Ezek a mérőszámok nem teljesen függetlenek egymástól, ahogy majd látni fogjuk.

Tiszta szemantika társítása a relációk attribútumaihoz

Amikor relációsémákba csoportosítjuk az attribútumokat, feltételezzük, hogy az egy relációhoz tartozó attribútumok egy bizonyos valós világbeli jelentéssel rendelkeznek, és megfelelő értelmezés társul hozzájuk. A relációk **szemantikája** a rekordokban megjelenő attribútumértékek értelmezését jelenti. Az 5. fejezetben megnéztük, hogy hogyan értelmezhetünk egy relációt tények egy halmazaként. Ha a 3. és 4. fejezetben leírt koncepcionális tervezést gondosan hajtjuk végre, és a 7. fejezetben leírt leképezési eljárást szisztematikusan követjük, akkor a relációséma-tervnek világos jelentéssel kell bírnia.

8.1. ábra - Egy egyszerűsített VÁLLALAT relációs adatbázisséma.



Általánosságban elmondható, hogy minél egyszerűbb megmagyarázni egy reláció szemantikáját, annál jobb a relációséma-terv. Ennek illusztrálására tekintsük a 8.1. ábrát, amelyen az 5.3. ábrán látható VÁLLALAT relációs adatbázisséma leegyszerűsített változata látható, valamint a 8.2. ábrát, amely e séma relációinak példa állapotait mutatja. A DOLGOZÓ relációséma jelentése roppant egyszerű: Minden rekord egy dolgozót reprezentál, egy-egy értékkel a dolgozó nevére (Dnév), személyi számára (Szsz), születési dátumára (Szdátum), címére (Lakcím), valamint azon osztály számára (Oszám) vonatkozóan, ahol a dolgozó dolgozik. Az Oszám attribútum külső kulcs, amely egy, a DOLGOZÓ és az OSZTÁLY közötti *implicit kapcsolatot* jelképez. Az OSZTÁLY és a PROJEKT sémák szemantikája szintén egyértelmű: Az OSZTÁLY rekordjai egy-egy osztály egyedet, a PROJEKT rekordjai pedig egy-egy projekt egyedet reprezentálnak. Az OSZTÁLY Ovez_szsz attribútuma összekapcsolja az osztályt azzal a dolgozóval, aki az adott osztály vezetője, a PROJEKT Oszám attribútuma pedig összekapcsolja a projektet az őt irányító osztállyal. Mindkettő külső kulcs attribútum. Az a könnyedség, amellyel a relációk attribútumainak a jelentését elmagyarázhatjuk, annak az *informális mérőszáma*, hogy milyen jól van megtervezve a reláció.

8.2. ábra - Példa adatbázis-állapot a 8.1. ábrán látható relációs adatbázissémához.

DOLGOZÓ

Dnév	Szsz	Szdátum	Lakcím	Oszám
Kovács László	1 650109 0812	1965. január 9.	4033 Debrecen	5
Szabó Mária	2 551208 2219	1955. december 8.	1097 Budapest	5
Kiss István	1 680119 6749	1968. január 19.	1172 Budapest	4
Takács József	1 410620 4902	1941. június 20.	4027 Debrecen	4
Horváth Erzsébet	2 620915 3134	1962. szeptember 15.	1092 Budapest	5
Tóth János	1 720731 2985	1972. július 31.	6726 Szeged	5
Fazekas Ilona	2 690329 1099	1969. március 29.	3535 Miskolc	4
Nagy Zoltán	1 371110 4519	1937. november 10.	1061 Budapest	1

OSZTÁLY

Onév	Oszám	Vez_szsz
Kutatás	5	2 551208 2219
Humán erőforrás	4	2 690329 1099
Központ	1	1 371110 4519

OSZT_HELYSZÍNEK

Oszám	Ohelyszín
1	Budapest
4	Kecskemét
5	Vác
5	Tiszafüred
5	Budapest

DOLGOZIK_RAJTA

Szsz	Pszám	Órák
1 650109 0812	1	32.5
1 650109 0812	2	7.5
2 620915 3134	3	40.0
1 720731 2985	1	20.0
1 720731 2985	2	20.0
2 551208 2219	2	10.0
2 551208 2219	3	10.0
2 551208 2219	10	10.0
2 551208 2219	20	10.0
1 680119 6749	30	30.0
1 680119 6749	10	10.0
2 690329 1099	10	35.0
2 690329 1099	30	5.0
1 410620 4902	30	20.0
1 410620 4902	20	15.0
1 371110 4519	20	NULL

PROJEKT

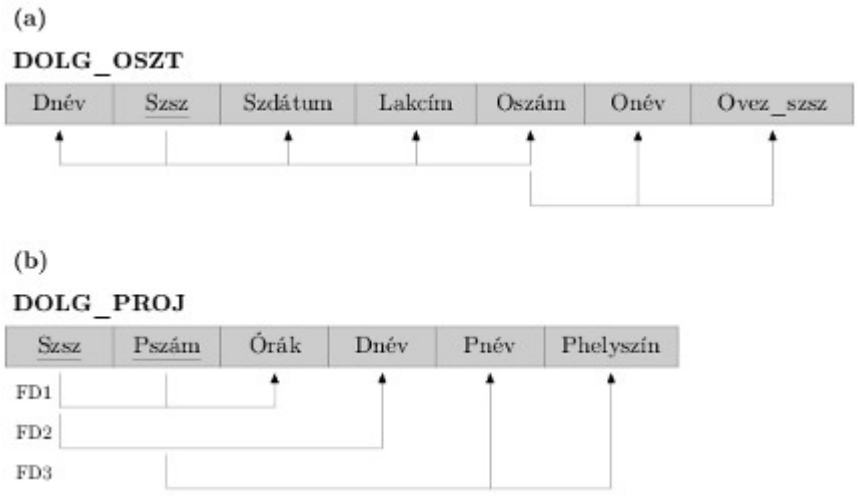
Pnév	Pszám	Phelyszín	Osz
X termék	1	Vác	5
Y termék	2	Tiszafüred	5
Z termék	3	Budapest	5
Komputerizáció	10	Kecskemét	4
Reorganizáció	20	Budapest	1
Új fejlesztések	30	Kecskemét	4

A 8.1. ábrán látható másik két relációséma szemantikája egy kicsit össszetettebb. A OSZT_HELYSZÍNEK rekordjai egy osztályszám (Oszám) mellett az adott osztály helyszíneinek *egyikét* (Ohelyszín) tartalmazzák. A DOLGOZIK_RAJTA rekordjai tárolják a dolgozó személyi számát (Szsz), azon projektek *egyikének* a számát, amelyeken az adott dolgozó dolgozik (Pszám), valamint azt az időtartamot, ahány órát az adott dolgozó hetente az adott projekten dolgozik (Órák). Mindkét sémának azonban jól definiált és egyértelmű értelmezése van. Az OSZT_HELYSZÍNEK séma az OSZTÁLY egy többértékű attribútumát reprezentálja, míg a DOLGOZIK_RAJTA egy M:N kapcsolatot ír le a DOLGOZÓ és a PROJEKT között. A 8.1. ábrán látható összes relációséma tehát könnyen magyarázhatónak, ezért a tiszta szemantika szempontjából jónak tekinthető. Ezek alapján a következő informális tervezési irányelvet fogalmazhatjuk meg.

1. irányelv. Úgy kell megtervezni a relációsémákat, hogy könnyű legyen megmagyarázni a

jelentésüket. Ne keverjük az attribútumokat több egyed- és kapcsolattípusból egy relációban. Intuitívan elmondható, hogy ha egy relációséma megfelel egy egyedtípusnak vagy egy kapcsolattípusnak, akkor könnyen értelmezhető és magyarázható a jelentése. Egyébként ha a reláció több egyed és kapcsolat keverékének felel meg, akkor az szemantikailag félreérthető lehet, és ezért a relációt nem lehet olyan könnyen magyarázni.

8.3. ábra - Két, módosítási anomáliákkal terhelt relációséma



A 8.3. (a) és (b) ábrán látható relációsémák szintén tiszta szemantikával rendelkeznek. (Az olvasó most figyelmen kívül hagyhatja a relációsémák alatti vonalakat; azok arra szolgálnak, hogy illusztrálják a funkcionális függések jelölését, amelyről a . alfejezetben lesz szó.) A 8.3. (a) ábrán látható DOLG_OSZT relációséma relációinak egy rekordja egy dolgozót reprezentál, de további információkat is tartalmaz — nevezetesen annak az osztálynak a nevét (Onév), ahol a dolgozó dolgozik, valamint az osztályvezető személyi számát (Ovez_szsz). A 8.3. (b) ábrán látható DOLG_PROJ relációséma relációinak egy rekordja összekapcsol egy dolgozót egy projekttel, de tartalmazza a dolgozó nevét (Dnév), a projekt nevét (Pnév) és a projekt helyszínét (Phelyszín) is. Bár ezzel a két relációsémával logikailag semmi gond nincs, hibás terveknek minősülnek, mert megsértik az 1. irányelvet azáltal, hogy különböző valós világbeli egyedek attribútumait keverik; a DOLG_OSZT a dolgozók és az osztályok, a DOLG_PROJ pedig a dolgozók, a projektek és a DOLGOZIK_RAJTA kapcsolat attribútumait keveri. Használhatók nézetként, de problémákat okozhat, ha alaprelációként használjuk őket, ahogy az a következő alfejezetből kiderül.

Redundáns információk a rekordokban és a karbantartási anomáliák

A sématervezés egyik célja, hogy minimalizálja az alaprelációk (és ezáltal a hozzájuk tartozó állományok) tárigényét. Az attribútumok relációsémákba történő csoportosításának jelentős hatása van a tárigényre. Hasonlítsuk össze például a 8.2. ábrán látható DOLGOZÓ és OSZTÁLY alaprelációk által elfoglalt tárhelyet a 8.4. ábrán látható DOLG_OSZT alaprelációéval, amely a DOLGOZÓ-ra és az OSZTÁLY-ra alkalmazott természetes összekapcsolás eredménye.

8.4. ábra - Példa állapotok a 8.2. ábra relációira alkalmazott természetes összekapcsolással kapott DOLG_OSZT és DOLG_PROJ relációkhoz. Ezek hatékonysági szempontok miatt alaprelációként tárolhatók.

DOLG_OSZT

Dnév	Szsz	Szdátum	Lakcím	Oszám	redundancia: Onév	redundancia: Ovez_szsz
Kovács László	1 650109 0812	1965. január 9.	4033 Debrecen	5	Kutatás	2 551208 2219
Szabó Mária	2 551208 2219	1955. december 8.	1097 Budapest	5	Kutatás	2 551208 2219
Kiss István	1 680119 6749	1968. január 19.	1172 Budapest	4	Humán erőforrás	2 690329 1099
Takács József	1 410620 4902	1941. június 20.	4027 Debrecen	4	Humán erőforrás	2 690329 1099
Horváth Erzsébet	2 620915 3134	1962. szeptember 15.	1092 Budapest	5	Kutatás	2 551208 2219
Tóth János	1 720731 2985	1972. július 31.	6726 Szeged	5	Kutatás	2 551208 2219
Fazekas Ilona	2 690329 1099	1969. március 29.	3535 Miskolc	4	Humán erőforrás	2 690329 1099
Nagy Zoltán	1 371110 4519	1937. november 10.	1061 Budapest	1	Központ	1 371110 4519

DOLG_PROJ

Szsz	Pszám	Órák	redundancia: Dnév	redundancia: Pnév	Phelyszín
1 650109 0812	1	32.5	Kovács László	X termék	Vác
1 650109 0812	2	7.5	Kovács László	Y termék	Tiszafüred
2 620915 3134	3	40.0	Horváth Erzsébet	Z termék	Budapest
1 720731 2985	1	20.0	Tóth János	X termék	Vác
1 720731 2985	2	20.0	Tóth János	Y termék	Tiszafüred
2 551208 2219	2	10.0	Szabó Mária	Y termék	Tiszafüred
2 551208 2219	3	10.0	Szabó Mária	Z termék	Budapest
2 551208 2219	10	10.0	Szabó Mária	Komputerizáció	Kecskemét
2 551208 2219	20	10.0	Szabó Mária	Reorganizáció	Budapest
1 680119 6749	30	30.0	Kiss István	Új fejlesztések	Kecskemét
1 680119 6749	10	10.0	Kiss István	Komputerizáció	Kecskemét
2 690329 1099	10	35.0	Fazekas Ilona	Komputerizáció	Kecskemét
2 690329 1099	30	5.0	Fazekas Ilona	Új fejlesztések	Kecskemét
1 410620 4902	30	20.0	Takács József	Új fejlesztések	Kecskemét
1 410620 4902	20	15.0	Takács József	Reorganizáció	Budapest
1 371110 4519	20	NULL	Nagy Zoltán	Reorganizáció	Budapest

A DOLG_OSZT-ban az egy konkrét osztályhoz (Oszám, Onév, Ovez_szsz) tartozó attribútumértékek *az adott osztályon dolgozó minden egyes alkalmazottnál* megismétlődnek. Ezzel szemben az egyes osztályokra vonatkozó információk csak egyszer jelennek meg a 8.2. ábra OSZTÁLY relációjában. Csak az osztály száma (Oszám) ismétlődik a DOLGOZÓ relációban külső kulcsként minden olyan dolgozónál, aki az adott osztályon dolgozik. Hasonló megjegyzéseket tehetünk a DOLG_PROJ relációra (lásd a 8.4. ábrát), amely kiegészíti a DOLGOZIK_RAJTA relációt a DOLGOZÓ és a PROJEKT egyes attribútumaival.

Egy másik súlyos probléma a 8.4. ábrán látható relációk alaprelációként történő felhasználásával a **karbantartási anomáliák** problémája. Ezeket csoportosíthatjuk beszúrási, törlési és módosítási anomáliákra.[27]

Beszúrási anomáliák. A beszúrási anomáliáknak két típusát különböztethetjük meg, amelyeket a következő példákkal szemléltetünk a DOLG_OSZT relációban:

- Ahhoz, hogy egy új dolgozó rekordot szúrjunk be a DOLG_OSZT-ba, vagy annak az osztálynak az attribútumértékeit kell megadni, ahol ez a dolgozó dolgozik, vagy pedig NULL-okat (ha a dolgozó még semelyik osztályon sem dolgozik). Ahhoz például, hogy beszúrjunk egy új rekordot egy olyan dolgozóval, aki az 5-ös számú osztályon dolgozik, úgy

kell megadnunk az 5-ös osztályhoz tartozó attribútumértékeket, hogy *konzisztensek* legyenek a DOLG_OSZT más rekordjaiban található, az 5-ös osztályhoz tartozó értékekkel. A 8.2. ábrán látható terv esetén nem kell aggódnunk emiatt a konzisztenciaprobléma miatt, mert a dolgozó rekordjában csak az osztály számát adjuk meg; az 5-ös osztályra vonatkozó minden más attribútumértéket csak egyszer szerepeltetünk az adatbázisban, az OSZTÁLY reláció rekordjaként.

- A DOLG_OSZT relációba nehézkes új osztály beszúrni, ahol még egyetlen dolgozó sem dolgozik. Ennek az egyetlen módja az, ha NULL értékeket írunk a dolgozó attribútumaihoz. Ez azért okoz problémát, mert az Szsz a DOLG_OSZT elsődleges kulcs, és minden rekordnak egy dolgozó egyedet kellene reprezentálnia, nem pedig egy osztály egyedet. Ráadásul amikor az első dolgozót hozzárendeljük ehhez az osztályhoz, onnantól kezdve nincs szükségünk erre a NULL értékeket tartalmazó rekordra. Ez a probléma nem lép fel a 8.2. ábrán látható terv esetén, mert az osztályt az OSZTÁLY relációba visszük be, akár dolgozik ott dolgozó, akár nem, és ha egy dolgozót hozzárendelünk ehhez az osztályhoz, akkor a megfelelő rekordot a DOLGOZÓ-ba szúrjuk be.

Törlési anomáliák. A törlési anomáliák a fent leírt második beszúrási anomáliához kapcsolódnak. Ha egy olyan dolgozó rekordot törlünk ki a DOLG_OSZT-ból, amely az utolsó, egy adott osztályon dolgozó alkalmazottat reprezentálja, akkor elveszítjük az adott osztályra vonatkozó információkat az adatbázisból. Ez a probléma nem merül fel a 8.2. ábrán látható adatbázisban, mert az OSZTÁLY rekordokat külön tároljuk.

Módosítási anomáliák. Ha a DOLG_OSZT-ban módosítjuk egy konkrét osztály valamely attribútumának az értékét (például az 5-ös számú osztály vezetőjét), akkor az összes olyan rekordot módosítanunk kell, amely az adott osztályon dolgozó alkalmazotthoz tartozik; máskülönben az adatbázis inkonzisztenssé válik. Ha nem módosítunk néhány rekordot, akkor ugyanazon osztályra vonatkozóan az osztályvezető személyi száma két különböző értéket fog fölvenni a különböző dolgozó rekordokban, és ez helytelen lenne.[28]

A felsorolt három anomália alapján felírhatjuk a következő irányelvünket.

2. irányelv. Az alaprelációk sémáit úgy kell megtervezni, hogy a relációkban ne forduljanak elő beszúrási, törlési és módosítási anomáliák. Ha mégis lennének anomáliák, akkor azokat világosan jelezzük, és győződjünk meg róla, hogy az adatbázist módosító programok helyesen működnek.

A második irányelv konzisztens az első irányelvvel, és annak egyfajta átfogalmazott változata. Láthatjuk azt is, hogy szükségünk van egy formálisabb megközelítésre ahhoz, hogy eldöntsük, hogy egy terv megfelel-e ezeknek az irányelveknek. A -től -ig tartó alfejezetek fogják megadni nekünk ezeket a formális fogalmakat. Fontos azonban megjegyeznünk, hogy ezeket az irányelveket néha *muszáj megsértenünk*, hogy *növeljük* bizonyos lekérdezések *hatékonyságát*. Ha például egy fontos lekérdezés egy dolgozó osztályáról és a dolgozó attribútumairól kér le információkat, akkor a DOLG_OSZT sémát használhatjuk alaprelációként. Ekkor azonban a DOLG_OSZT-ban lévő anomáliákat figyelembe kell vennünk és számolnunk kell velük úgy, hogy amennyiben az alapreláció módosul, ne maradjon inkonzisztencia az adatbázisban (például triggerek vagy tárolt eljárások használatával, amelyek automatikus módosításokat hajtanak végre). Általánosságban tanácsos anomáliamentes alaprelációkat használni, és nézeteket megadni a fontos lekérdezésekben gyakran hivatkozott attribútumok összekapcsolására. Ezzel csökkenthetjük a lekérdezésekben szereplő összekapcsolási műveletek számát, egyszerűbbé téve ezzel a lekérdezés helyes felírását, ráadásul sok esetben ez a hatékonyságot is növeli.[29]

NULL értékek a rekordokban

Bizonyos sématervekben sok attribútumot csoportosítunk egy „kövér” relációba. Ha sok olyan attribútum van, amely nem értelmezhető a reláció minden rekordjában, akkor azokban a

rekordokban rengeteg NULL értéket kapunk. Ez a tárolási szinten helypazarlást jelent, és problémát fog okozni az attribútumok jelentésének a megértése, illetve logikai szinten az összekapcsolások megadása.[30] Egy másik probléma a NULL-okkal, hogy hogyan kezeljük őket az olyan csoportosító műveletekkel, mint például a COUNT vagy a SUM. A szelekció és az összekapcsolás művelete összehasonlításokat tartalmaznak. Ha NULL értékek vannak jelen, az eredmény megjósolhatatlan lesz. Ráadásul a NULL-ok többféle értelmezésben szerepelhetnek, amelyek a következők:

- Az attribútum *nem értelmezhető* ebben a rekordban.
- *Nem tudjuk*, hogy az attribútum értéke ebben a rekordban *létezik-e*.
- Az érték *létezik, de hiányzik*, azaz még nem rögzítettük.

Mivel minden NULL-t ugyanúgy reprezentálunk, ezzel összemossuk a lehetséges különböző jelentéseit. Ezért egy újabb irányelvet fogalmazhatunk meg.

3. irányelv. Amennyire csak lehetséges, ne tegyünk az alaprelációkba olyan attribútumokat, amelyek gyakran vesznek fel NULL értéket. Ha a NULL-ok elkerülhetetlenek, akkor győződjünk meg róla, hogy csak kivételes esetekben alkalmazzuk, nem pedig a reláció rekordjainak többségében.

A tárhely hatékony kihasználása és az összekapcsolások elkerülése az a két egymásnak ellentmondó követelmény, amelyek meghatározzák, hogy bevegyük-e a NULL értékeket tartalmazó oszlopokat a relációba, vagy külön relációt hozzunk létre ezen oszlopok számára (a megfelelő kulcs oszlopokkal együtt). Ha például csak a dolgozók 10%-ának van külön irodája, akkor nehéz megindokolni, hogy bevegyünk egy Szobaszám attribútumot a DOLGOZÓ relációba; ehelyett inkább egy DOLG_SZOBÁK(Dszsz, Szobaszám) relációt hozhatunk létre, kizárólag a külön irodával rendelkező dolgozókhoz tartozó rekordokkal.

Álrekordok generálása

Tekintsük a 8.5. (a) ábrán látható DOLG_HELYSZÍNEK és DOLG_PROJ1 relációsémákat, amelyeket a 8.3. (b) ábrán látható egyetlen DOLG_PROJ reláció helyett használhatunk. A DOLG_HELYSZÍNEK egy rekordja azt jelenti, hogy a Dnév nevű alkalmazott *egy olyan projekten* dolgozik, amelynek a helyszíne Phelyszín. A DOLG_PROJ1 egy rekordja azt a tényt írja le, hogy az Szsz személyi számú dolgozó Órák órát dolgozik hetente azon a projekten, amelynek a neve, száma és helyszíne Pnév, Pszám és Phelyszín. A 8.5. (b) ábra a DOLG_HELYSZÍNEK és a DOLG_PROJ1 relációállapotait mutatja, amelyek a 8.4. ábrán látható DOLG_PROJ relációnak felelnek meg, és amelyeket a DOLG_PROJ-ra alkalmazott megfelelő projekció (π) műveleteket alkalmazva kapunk meg (egyelőre tekintsünk el a 8.5. (b) ábrán látható szaggatott vonalaktól).

8.5. ábra - A 8.3. (b) ábra DOLG_PROJ relációjának különösen rossz terve. (a) A DOLG_HELYSZÍNEK és a DOLG_PROJ1 relációsémák. (b) A 8.4. ábra DOLG_PROJ relációjának a DOLG_HELYSZÍNEK és DOLG_PROJ1 relációkra történő vetítésének az eredménye.

(a)

DOLG_HELYSZÍNEK

Dnév	Phelyszín

P.K.

DOLG_PROJ1

Szsz	Pszám	Órák	Pnév	Phelyszín

P.K.

(b)

DOLG_HELYSZÍNEK

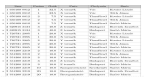
Dnév	Phelyszín
Kovács László	Vác
Kovács László	Tiszafüred
Horváth Erzsébet	Budapest
Tóth János	Vác
Tóth János	Tiszafüred
Szabó Mária	Tiszafüred
Szabó Mária	Budapest
Szabó Mária	Kecskemét
Kiss István	Kecskemét
Fazekas Ilona	Kecskemét
Takács József	Kecskemét
Takács József	Budapest
Nagy Zoltán	Budapest

DOLG_PROJ1

Szsz	Pszám	Órák	Pnév	Phelyszín
1 650109 0812	1	32.5	X termék	Vác
1 650109 0812	2	7.5	Y termék	Tiszafüred
2 620915 3134	3	40.0	Z termék	Budapest
1 720731 2985	1	20.0	X termék	Vác
1 720731 2985	2	20.0	Y termék	Tiszafüred
2 551208 2219	2	10.0	Y termék	Tiszafüred
2 551208 2219	3	10.0	Z termék	Budapest
2 551208 2219	10	10.0	Komputerizáció	Kecskemét
2 551208 2219	20	10.0	Reorganizáció	Budapest
1 680119 6749	30	30.0	Új fejlesztések	Kecskemét
1 680119 6749	10	10.0	Komputerizáció	Kecskemét
2 690329 1099	10	35.0	Komputerizáció	Kecskemét
2 690329 1099	30	5.0	Új fejlesztések	Kecskemét
1 410620 4902	30	20.0	Új fejlesztések	Kecskemét
1 410620 4902	20	15.0	Reorganizáció	Budapest
1 371110 4519	20	NULL	Reorganizáció	Budapest

Tételezzük fel, hogy a DOLG_PROJ helyett a DOLG_PROJ1-et és a DOLG_HELYSZÍNEK-et használjuk alaprelációkként. Ez egy különösen rossz sématervet eredményez, mivel a DOLG_PROJ1-ből és a DOLG_HELYSZÍNEK-ből nem tudjuk visszanyerni azokat az információkat, amelyek eredetileg a DOLG_PROJ-ban voltak. Ha megkísérlünk végrehajtani egy természetes összekapcsolást a DOLG_PROJ1 és a DOLG_HELYSZÍNEK relációkra, akkor az eredmény sokkal több rekordot fog tartalmazni, mint amennyi eredetileg a DOLG_PROJ-ban volt. A 8.6. ábrán csak a 8.5. (b) ábra szaggatott vonalai fölötti rekordokra alkalmazott összekapcsolás eredményét láthatjuk (hogy csökkentsük az eredményül kapott reláció méretét). Azokat az extra rekordokat, amelyek nem szerepltek a DOLG_PROJ-ban, **álrekordoknak** nevezzük, mert érvénytelen, hamis információkat reprezentálnak. A 8.6. ábrán csillaggal (*) jelöltük az álrekordokat.

8.6. ábra - A 8.5. ábra DOLG_PROJ1 és DOLG_HELYSZÍNEK relációinak a szaggatott vonalak fölötti rekordjaira alkalmazott természetes összekapcsolás eredménye. A kapott álrekordokat csillaggal jelöltük meg.



A DOLG_PROJ felbontása a DOLG_HELYSZÍNEK-re és a DOLG_RPOJ1-re nem célszerű, mert ha összekapcsoljuk őket egy természetes összekapcsolással, akkor nem kapjuk vissza az eredeti

helyes információkat. Ez azért van, mert ebben az esetben a Phelyszín az az attribútum, amely összekapcsolja a DOLG_HELYSZÍNEK-et és DOLG_PROJ1-et, és a Phelyszín nem elsődleges kulcs és nem is külső kulcs sem a DOLG_HELYSZÍNEK-ben, sem a DOLG_PROJ1-ben. Ezek után informálisan kimondhatunk egy újabb tervezési irányelvet.

4. irányelv. Úgy kell megtervezni a relációsémákat, hogy oly módon lehessen őket összekapcsolni (elsődleges kulcs, kulső kulcs) attribútumpárok egyenlősége alapján, amely garantálja, hogy nem keletkeznek álrekordok. Kerüljük azokat a relációkat, amelyek olyan, azonos szerepű attribútumokat tartalmaznak, amik nem (külső kulcs, elsődleges kulcs) kombinációk, mert az ilyen attribútumok alapján történő összekapcsolás álrekordokat eredményezhet.

Ezt az informális irányelvet nyilván formálisabban is meg kell fogalmazni. A 9. fejezetben ismertetünk egy formális feltételt, amit nemadditív (vagy veszteségmentes) join tulajdonságnak hívunk, és amely garantálja, hogy bizonyos összekapcsolások nem állítanak elő álrekordokat.

A tervezési irányelvek összegzése

A -től -ig terjedő alfejezetekben informálisan ismertettünk olyan szituációkat, amelyek problémás relációsémákhoz vezetnek, és informális irányelveket javasoltunk a jó relációs tervezéshez. A bemutatott problémák, amelyek további elemző eszközök nélkül is felismerhetők, az alábbiak:

- anomáliák, amelyek felesleges munkát okoznak egy relációba való beszúrás, illetve egy reláció módosítása folyamán, és amelyek információk véletlen elvesztését okozhatják egy relációból való törlés folyamán;
- tárpazarlás, valamint a szelekciós, a csoportosítási és az összekapcsolási műveletek elvégzésének a nehézsége a NULL értékeknek köszönhetően;
- érvénytelen és hamis adatok előállítása nem megfelelően kapcsolódó alaprelációk összekapcsolása során.

A fejezet további részében olyan formális fogalmakat és elméletet ismertetünk, amelyeket arra használhatunk, hogy segítségükkel *különálló* relációsémák *jóságát* és *rosszaságát* precízebben definiáljuk. Először a funkcionális függést mint az elemzés egy eszközét ismertetjük. Ezután a relációsémákra vonatkozó három normálformáról és a Boyce–Codd-normálformáról írunk. A 9. fejezetben további normálformákat definiálunk, amelyek az adatfüggés újabb típusain, a többértékű függésen és a kapcsolásfüggésen alapulnak.

Funkcionális függések

Az egyik legfontosabb fogalom a relációs sématervezés elméletében a funkcionális függés fogalma. Ebben a fejezetben definiálni fogjuk a funkcionális függés fogalmát, a . alfejezetben pedig megmutatjuk, hogyan lehet felhasználni ezt a fogalmat relációs sémák normálformáinak a definiálására.

A funkcionális függés definíciója

A funkcionális függés egy olyan megszorítás, amely az adatbázis két attribútumhalmaza között áll fenn. Tegyük fel, hogy a relációs adatbázissémánknak n attribútuma van: $A_1, A_2, ..., A_n$; és gondoljunk az egész adatbázisunkra úgy, hogy azt egyetlen, **univerzális** $R = \{ A_1, A_2, \ldots, A_n \}$ relációsémával írjuk le. Ez nem jelenti azt, hogy az adatbázisunkat egyetlen univerzális táblaként fogjuk tárolni; ezt a fogalmat csak az adatfüggőség formális elméletének a kialakításához használjuk.

Definíció. Az R két attribútumhalmaza, X és Y között értelmezett, $X \rightarrow Y$ alakú **funkcionális**

függés egy *megszorítást* ír elő az *R* relációséma bármely *r* relációjának lehetséges rekordjaira. A megszorítás az, hogy bármely két *r*-beli t_1 és t_2 rekord esetén, amelyekre $t_1[X] = t_2[X]$ teljesül, teljesülnie kell $t_1[Y] = t_2[Y]$-nak is.

Ez azt jelenti, hogy *r*-beli rekord *Y* komponensének az értékei *függnek* az *X* komponens értékektől; más szavakkal egy rekord *X* komponensének az értékei egyértelműen (vagy **funkcionálisan**) *meghatározzák* az *Y* komponens értékeit. Azt is mondhatjuk, hogy *X* **funkcionálisan meghatározza** *Y*-t, vagy *Y* **funkcionálisan függ** *X*-től. Az *X* attribútumhalmazt a funkcionális függés **bal oldalának**, az *Y*-t pedig a **jobb oldalának** nevezzük.

Ennélfogva *X* akkor és csak akkor határozza meg funkcionálisan *Y*-t egy *R* relációsémában, ha valahányszor *r*(*R*) két rekordja megegyezik az *X*-hez tartozó értékeikben, szükségszerűen megegyeznek az *Y*-hoz tartozó értékeikben is. Vegyük észre a következőket:

- Abból, hogy egy *R*-re előírt megszorítás szerint bármely *r*(*R*) relációpéldányban nem szerepelhet több rekord egy adott *X*-hez tartozó értékkel — azaz *X* **kulcsjelöltje** *R*-nek —, következik $X \rightarrow Y$ az *R* attribútumainak bármely *Y* részhalmazára (mivel a kulcsmegszorításból következik, hogy egyetlen legális *r*(*R*) állapotban sem lehet két olyan rekord, amelyeknek azonosak lennének az *X*-hez tartozó értékeik).
- Ha $X \rightarrow Y$ teljesül *R*-ben, még semmit sem tudunk mondani arról, hogy vajon $Y \rightarrow X$ is teljesül-e *R*-ben.

Egy funkcionális függés **az attribútumok szemantikájának** vagy **jelentésének** egy jellemzője. Az adatbázistervezők *R* attribútumai szemantikájának — azaz hogy milyen kapcsolat van közöttük — az általuk vett értelmezését használják fel ahhoz, hogy megadják azokat a funkcionális függéseket, amelyek *R minden r* relációállapotában fennállnak. Valahányszor *R* két attribútumhalmazának a jelentése azt sugallja, hogy egy funkcionális függés fennáll, akkor felvesszük ezt a függést a megszorítások közé. Azokat a *r*(*R*) relációállapotokat, amelyek megfelelnek a funkcionális függés megszorításoknak, *R* **legális** vagy **jogszerű relációállapotainak** nevezzük. A funkcionális függések fő felhasználása tehát az, hogy tovább jellemezzük az *R* relációsémát azzal, hogy az attribútumaira olyan megszorításokat adunk, amelyek *mindig* érvényesek. Bizonyos funkcionális függéseket anélkül is felírhatunk, hogy egy konkrét relációra utalnánk, ehelyett a szóban forgó attribútumok egy tulajdonságaként adjuk meg őket a hétköznapi értelmük alapján. Például az {Állam, Jogosítványszám} → Társadalombiztosítási_szám fennáll minden egyesült államokbeli felnőtt esetén. Az is elképzelhető, hogy egyes funkcionális függések a valós világban megszűnnek létezni, ha a kapcsolat módosul. Például az Irányítószám → Körzetszám valaha létező kapcsolat volt a postai kódok és a telefonszámkódok között az Egyesült Államokban, de a körzetszámok elburjánzása miatt ez ma már nem teljesül.

Tekintsük a 8.3. (b) ábrán szereplő DOLG_PROJ relációsémát; az attribútumok jelentése alapján tudjuk, hogy a következő funkcionális függéseknek kell fennállniuk:

a. Szsz → Dnév
b. Pszám → {Pnév, Phelyszín}
c. {Szsz, Pszám} → Órák

Ezek a funkcionális függések azt mutatják, hogy (a) egy dolgozó személyi száma (Szsz) egyértelműen meghatározza a dolgozó nevét (Dnév), (b) a projektszámnak egy értéke (Pszám) egyértelműen meghatározza a projekt nevét (Pnév) és helyszínét (Phelyszín), és (c) az Szsz és a PSzám értékeinek együttese egyértelműen meghatározza azt az időtartamot, ahány órát jelenleg a dolgozó a projekten hetente dolgozik (Órák). Másképpen úgy is mondhatjuk, hogy a Dnév funkcionálisan függ az Szsz-től (az Szsz funkcionálisan meghatározza a Dnév attribútumot), vagy *az Szsz egy adott értéke esetén ismerjük a Dnév értékét*, stb.

8.7. ábra - A TANÍT egy relációállapota a *lehetséges* Jegyzet → Kurzus funkcionális függéssel.

Az Oktató → Kurzus azonban ki van zárva.

TANÍT

Oktató	Kurzus	Jegyzet
Fazekas	Adatszerkezetek	Bartram
Fazekas	Adatkezelés	Martin
Asszalós	Fordítóprogramok	Hoffman
Pánovics	Adatszerkezetek	Horowitz

A funkcionális függés *az R relációséma egy tulajdonsága*, nem az *R* egy konkrét *r* legális relációállapotáé. A funkcionális függések ezért *nem* következnek automatikusan egy adott *r* relációállapotból, hanem explicit módon kell definiálnia valaki olyan személynek, aki ismeri *R* attribútumainak szemantikáját. Példaként tekintsük a 8.7. ábrán a TANÍT relációséma egy konkrét állapotát. Bár első pillantásra úgy tűnhet, hogy Jegyzet → Kurzus, ezt nem tudjuk megerősíteni, hacsak nem tudjuk, hogy a TANÍT *minden lehetséges legális állapotára* érvényes. Elegendő azonban *egyetlen ellenpélda* annak demonstrálására, hogy megcáfoljunk egy funkcionális függést. Például abból, hogy „Fazekas" mind „Adatszerkezeteket", mind pedig „Adatkezelést" tanít, következtethetünk, hogy az Oktató nem határozza meg funkcionálisan a Kurzust.

A 8.3. ábra egy **grafikus jelölést** vezet be a funkcionális függések ábrázolására: Minden funkcionális függést egy vízszintes vonal jelöl. A funkcionális függés bal oldali attribútumait függőleges vonalak kapcsolják a függést jelképező vonalhoz, míg a jobb oldali attribútumokat az attribútumok felé mutató nyilak kötik össze vele, ahogy a 8.3. (a) és (b) ábrákon látható.

A funkcionális függések levezetési szabályai

Jelöljük *F*-fel az *R* relációséma feletti funkcionális függések halmazát. A sématervező tipikusan csak a *szemantikailag nyilvánvaló* funkcionális függéseket adja meg; általában azonban számos további funkcionális függés is fennáll különböző attribútumhalmazok között *minden* legális relációpéldány esetében, amelyek következnek az *F*-beli függésekből. Ezek a további függések *levezethetők* vagy *következnek* az *F*-beli funkcionális függésekből.

A valós életben lehetetlen megadni az összes lehetséges funkcionális függést egy konkrét esetben. Ha például minden osztálynak egy vezetője van, tehát Oszám egyértelműen meghatározza Ovez_szsz-t (Oszám → Ovez_szsz), és az osztályvezetőnek egyetlen telefonszáma van, amelyet Ovez_telefonnak hívunk (Ovez_szsz → Ovez_telefon), akkor ebből a két függésből együtt következik, hogy Oszám → Ovez_telefon. Ez egy levezetett funkcionális függés, és *nem* kell explicit módon megadni a két említett funkcionális függés mellett. Formálisan tehát célszerű definiálni egy olyan fogalmat — amit *lezártnak* nevezünk —, amely az *F* halmazból levezethető összes lehetséges függést tartalmazza.

Definíció. Formálisan azt a függéshalmazt, amely tartalmazza az összes *F*-beli függést éppúgy, mint azokat a függéseket, amelyek levezethetők *F*-ből, *F* **lezártjának** nevezzük és F^+-szal jelöljük.

Az $X \rightarrow Y$ funkcionális függés **levezethető** az *R*-re vonatkozó *F* függéshalmazból, ha $X \rightarrow Y$ teljesül *R* minden *r* relációjában; azaz ha valahányszor *r* kielégíti az *F*-ben lévő összes függést, mindannyiszor $X \rightarrow Y$ is teljesül *r*-ben. Az *F* függéshalmaz F^+ lezártja az összes, *F*-ből levezethető funkcionális függés halmaza. Ahhoz, hogy megadjuk a függések levezetésének egy szisztematikus módszerét, **levezetési szabályok** egy halmazát kell bevezetnünk, amelyek segítségével új függéseket vezethetünk le egy adott függéshalmazból. A következőkben ilyen levezetési szabályok közül tekintünk át néhányat. Az $F \models X \rightarrow Y$ jelölést fogjuk használni annak jelölésére, hogy az *F* függéshalmazból az $X \rightarrow Y$ funkcionális függés levezethető.

A következőkben rövidített jelölést fogunk használni a funkcionális függések tárgyalásánál. A kényelem kedvéért az attribútumváltozókat konkatenáljuk, a vesszőket pedig elhagyjuk. Így az $\{ X, Y \} \rightarrow Z$ funkcionális függést röviden $XY \rightarrow Z$, míg az $\{ X, Y, Z \} \rightarrow \{ U, V \}$ funkcionális függést $XYZ \rightarrow UV$ alakban írjuk. A következő hat szabály a funkcionális függések jól ismert levezetési szabályai:

1. **A reflexivitás szabálya.** Ha $X \supseteq Y$, akkor $X \rightarrow Y$.

2. **Az augmentivitás szabálya.** $\{ X \to Y \} \models XZ \to YZ$.
3. **A tranzitivitás szabálya.** $\{ X \to Y, Y \to Z \} \models X \to Z$.
4. **A dekompozíció szabálya.** $\{ X \to YZ \} \models X \to Y$.
5. **Az additivitás vagy unióképzés szabálya.** $\{ X \to Y, X \to Z \} \models X \to YZ$.
6. **A pszeudotranzitivitás szabálya.** $\{ X \to Y, WY \to Z \} \models WX \to Z$.

A reflexivitás szabálya kimondja, hogy attribútumok egy halmaza mindig meghatározza saját magát vagy önmaga egy részhalmazát, ami nyilvánvaló. Mivel ez a szabály olyan függéseket generál, amelyek mindig igazak, az ilyen szabályokat *triviálisnak* nevezzük. Formálisan: egy $X \to Y$ függés **triviális**, ha $X \supseteq Y$; egyébként **nemtriviális**. Az augmentivitás szabálya azt állítja, hogy egy függés mindkét oldalához ugyanazt az attribútumhalmazt hozzáadva újabb érvényes függést kapunk. A tranzitivitás szabálya szerint a funkcionális függések tranzitívak. A dekompozíció szabálya kimondja, hogy egy függés jobb oldaláról elhagyhatunk attribútumokat; ezt a szabályt egymás után többször alkalmazva az $X \to \{ A_1, A_2, \ldots, A_n \}$ funkcionális függés felbontható az $\{ X \to A_1, X \to A_2, \ldots, X \to A_n \}$ függéshalmazra. Az unióképzés szabálya ennek az ellenkezőjét teszi lehetővé; funkcionális függések egy $\{ X \to A_1, X \to A_2, \ldots, X \to A_n \}$ halmazát összevonhatjuk egyetlen $X \to \{ A_1, A_2, \ldots, A_n \}$ funkcionális függéssé.

Ezeket a szabályokat nagy körültekintéssel kell alkalmazni. Bár $X \to A$-ból és $X \to B$-ből következik $X \to AB$ a fent említett unióképzés szabálya alapján, azonban önmagában sem $X \to A$-ból, sem $Y \to B$-ből *nem* következik $XY \to AB$. Hasonlóan $XY \to A$-ból *sem* következik szükségszerűen $X \to A$ és $Y \to A$.

A felsorolt levezetési szabályok mindegyike bebizonyítható a funkcionális függés definíciója alapján vagy közvetlenül, vagy **indirekt módon**. Az indirekt módon történő bizonyítás során feltételezzük, hogy a szabály nem teljesül, és megmutatjuk, hogy ez nem lehetséges. A következőkben bebizonyítjuk, hogy az első három szabály érvényes. A másodikat indirekt módon bizonyítjuk.

A reflexivitás bizonyítása. Tegyük fel, hogy $X \supseteq Y$ és hogy léteznek t_1 és t_2 rekordok az R séma valamely r relációjában, melyekre $t_1[X] = t_2[X]$. Ekkor $t_1[Y] = t_2[Y]$ is teljesül, mivel $X \supseteq Y$. Tehát $X \to Y$ biztosan teljesül r-en.

Az augmentivitás bizonyítása (indirekt módon). Tegyük fel, hogy $X \to Y$ teljesül R egy r relációpéldányában, de $XZ \to YZ$ nem teljesül benne. Ekkor kell léteznie két rekordnak, t_1-nek és t_2-nek r-ben úgy, hogy (1) $t_1[X] = t_2[X]$, (2) $t_1[Y] = t_2[Y]$, (3) $t_1[XZ] = t_2[XZ]$ és (4) $t_1[YZ] \neq t_2[YZ]$. Ez nem lehetséges, mert (1)-ből és (3)-ból levezethetjük, hogy (5) $t_1[Z] = t_2[Z]$, míg (2)-ből és (5)-ből következik (6) $t_1[YZ] = t_2[YZ]$, ami ellentmond (4)-nek.

A tranzitivitás bizonyítása. Tegyük fel, hogy mind (1) $X \to Y$, mind pedig (2) $Y \to Z$ teljesül egy r relációban. Ekkor bármely két r-beli t_1 és t_2 rekordra, amelyre $t_1[X] = t_2[X]$ teljesül, az (1)-es feltevés miatt kapjuk, hogy (3) $t_1[Y] = t_2[Y]$; majd a (3)-ból és a (2)-es feltevésből azt is kapjuk, hogy (4) $t_1[Z] = t_2[Z]$; így teljesülnie kell $X \to Z$-nek r-ben.

Hasonló bizonyítási lépéseket felhasználva bebizonyíthatjuk a dekompozíció, az additivitás és a pszeudotranzitivitás szabályát, valamint tetszőleges további érvényes levezetési szabályokat is. Mindazonáltal sokkal egyszerűbb úgy bebizonyítani egy levezetési szabályról azt, hogy érvényes a funkcionális függésekre, ha azoknak a levezetési szabályoknak a felhasználásával tesszük meg ezt, amelyekről korábban már bebizonyítottuk, hogy érvényesek. A példa kedvéért a dekompozíció, az additivitás és a pszeudotranzitivitás szabályát a következőképpen bizonyíthatjuk be a reflexivitás, az augmentivitás és a tranzitivitás szabályainak a felhasználásával:

A dekompozíció bizonyítása (a reflexivitás, az augmentivitás és a tranzitivitás felhasználásával).

1. $X \to YZ$ (adott).
2. $YZ \to Y$ (tudjuk, hogy $YZ \supseteq Y$, amire alkalmazzuk a reflexivitás szabályát).
3. $X \to Y$ (1-re és 2-re alkalmazzuk a tranzitivitás szabályát).

Az additivitás vagy unióképzés bizonyítása (a reflexivitás, az augmentivitás és a tranzitivitás felhasználásával).

1. $X \to Y$ (adott).
2. $X \to Z$ (adott).
3. $X \to XY$ (1-re alkalmazzuk az augmentivitás szabályát X-szel; megjegyezve, hogy $XX = X$).
4. $XY \to YZ$ (2-re alkalmazzuk az augmentivitás szabályát Y-nal).
5. $X \to YZ$ (3-ra és 4-re alkalmazzuk a tranzitivitás szabályát).

A pszeudotranzitivitás bizonyítása (a reflexivitás, az augmentivitás és a tranzitivitás felhasználásával).

1. $X \to Y$ (adott).
2. $WY \to Z$ (adott).
3. $WX \to WY$ (1-re alkalmazzuk az augmentivitás szabályát W-vel).
4. $WX \to Z$ (3-ra és 2-re alkalmazzuk a tranzitivitás szabályát).

Armstrong 1974-ben megmutatta, hogy a reflexivitás, az augmentivitás és a tranzitivitás szabálya együtt helyes és teljes. **Helyesség** alatt azt értjük, hogy ha adott egy R relációsémán fennálló funkcionális függéseknek egy F halmaza, akkor bármilyen függés, amely levezethető F-ből a három szabály segítségével, teljesülni fog R minden olyan r relációjában, amely *kielégíti az* F-beli *függéseket*. **Teljesség** alatt azt értjük, hogy a három szabályt mindaddig ismételten alkalmazva, míg már nem kapunk újabb függéseket, előállítható az F-ből levezethető *összes lehetséges függés* teljes halmaza. Más szavakkal, a függések F^+ halmaza, amelyet F **lezártjának** nevezünk, meghatározható F-ből kiindulva, kizárólag a három szabály alkalmazásával. A reflexivitás, az augmentivitás és a tranzitivitás szabályát együtt **Armstrong-axiómáknak** is nevezzük.

Egy adatbázis-tervező először általában a funkcionális függések azon F halmazát adja meg, amely könnyen meghatározható R attribútumainak a szemantikájából; azután a reflexivitás, augmentivitás és tranzitivitás szabályát felhasználva további funkcionális függéseket vezet le, amelyek szintén teljesülni fognak R-en. Ezen funkcionális függések meghatározásának egy szisztematikus módja az, ha először meghatározzuk azon X attribútumhalmazok mindegyikét, amelyek megjelennek valamely F-beli funkcionális függés bal oldalán, majd meghatározzuk az *összes olyan attribútum* halmazát, amelyek függnek X-től.

Definíció. Minden ilyen X attribútumhalmaz esetén meghatározzuk az attribútumoknak azt az X^+ halmazát, amelyek funkcionálisan függnek X-től az F alapján; X^+-t ***X F* feletti lezártjának** nevezzük. X^+ kiszámítására az 1. algoritmus használható.

1. algoritmus − *X F* feletti lezártjának, X^+-nak a meghatározása.

- $X^+ := X$;
- **repeat**
    - $oldX^+ := X^+$;

        - **for each** F-beli $Y \rightarrow Z$ funkcionális függésre **do**
            - **if** $X^+ \supseteq Y$ **then** $X^+ := X^+ \cup Z$;
    - **until** $(X^+ = oldX^+)$;

A 1. algoritmus azzal kezdődik, hogy X^+-nak értékül adja az X-et. Az IR1 szabály alapján tudjuk, hogy ezen attribútumok mindegyike funkcionálisan függ X-től. Az IR3 és IR4 szabályok felhasználásával újabb attribútumokat adunk hozzá X^+-hoz az F-beli funkcionális függések alapján. Végigmegyünk az összes F-beli függésen (a **repeat** ciklusban), amíg már nem tudunk újabb attribútumot hozzáadni X^+-hoz az F-beli függéseket bejáró (**for**) ciklus teljes lefutása során. Vegyük például a 8.3. ábra DOLG_PROJ relációsémáját; az attribútumok szemantikájából a DOLG_PROJ sémán fennálló funkcionális függések következő F halmazát adhatjuk meg:

- F = { Szsz, → Dnév, Pszám → { Pnév, Phelyszín }, { Szsz, Pszám } → Órák }

A 1. algoritmus segítségével az alábbi lezárt halmazokat állítjuk elő F-re vonatkozóan:

- { Szsz }$^+$ = { Szsz, Dnév }
- { Pszám }$^+$ = { Pszám, Pnév, Phelyszín }
- { Szsz, Pszám }$^+$ = { Szsz, Pszám, Dnév, Pnév, Phelyszín, Órák }

A jobb oldalon álló attribútumok halmaza a fenti sorokban azokat az attribútumokat tartalmazza, amelyek a bal oldalon álló attribútumhalmaztól funkcionálisan függnek a megadott F halmaz alapján.

Funkcionális függések halmazainak ekvivalenciája

A fejezet további részeiben funkcionális függések két halmazának az ekvivalenciájával foglalkozunk. Ehhez először is néhány előzetes definíciót adunk.

Definíció. Azt mondjuk, hogy funkcionális függések egy F halmaza **lefedi** funkcionális függések egy másik, E halmazát, ha minden E-beli funkcionális függés F^+-ban is fennáll; azaz ha minden E-beli függés levezethető F-ből.

Definíció. Funkcionális függések E és F halmazai ekvivalensek, ha $E^+ = F^+$. Az ekvivalencia tehát azt jelenti, hogy minden E-beli funkcionális függés levezethető F-ből, és minden F-beli funkcionális függés levezethető E-ből; azaz E akkor ekvivalens F-fel, ha E lefedi F-et, *és* F is lefedi E-t.

Azt, hogy F lefedi-e E-t, úgy dönthetjük el, hogy kiszámítjuk X^+-t F-*re nézve* minden E-beli $X \rightarrow Y$ funkcionális függésre, majd ellenőrizzük, hogy X^+ tartalmazza-e az Y-beli attribútumokat. Ha ez *minden* E-beli funkcionális függésre igaz, akkor F lefedi E-t. Azt, hogy E és F ekvivalensek-e, úgy döntjük el, hogy megvizsgáljuk, hogy E is lefedi-e F-et, és F is lefedi-e E-t.

Funkcionális függések minimális halmaza

Funkcionális függések egy E halmazának a **minimális lefedése** informálisan funkcionális függéseknek egy olyan F halmaza, amelyre teljesül, hogy minden E-beli függés az F-nek az F^+ lezártjában is megtalálható. Ráadásul ez a tulajdonság nem teljesül, ha F-ből bármelyik függést kivesszük; F-ben nem lehet redundancia, ugyanakkor az F-beli függések standard formájúak. Hogy eleget tegyünk ezeknek a tulajdonságoknak, formálisan akkor mondjuk funkcionális függések egy F halmazára, hogy **minimális**, hogyha eleget tesz a következő feltételeknek:

1. Az összes F-beli függés jobb oldalán egyetlen attribútum áll.

2. Egyetlen F-beli $X \to A$ függést sem cserélhetünk ki olyan $Y \to A$ függésre, ahol Y valódi részhalmaza X-nek, és az így kapott függéshalmaz ekvivalens F-fel.
3. Nem távolíthatunk el egyetlen függést sem F-ből úgy, hogy a kapott függéshalmaz ekvivalens marad F-fel.

A függések minimális halmazát úgy képzelhetjük el, mint *standard* vagy *kanonikus alakú* függések *redundanciáktól mentes* halmazát. Az első feltétel csak annyit mond ki, hogy minden függésnek kanonikus alakban kell lennie, azaz a jobb oldalon csak egyetlen attribútum állhat.[31] A második és harmadik feltételek biztosítják, hogy ne legyen redundancia a függések között, amely redundanciát vagy az okozhatja, hogy egy függés bal oldalán redundáns attribútumok állnak (2. feltétel), vagy az, hogy olyan függés van F-ben, amely a többi F-beli függésből levezethető (3. feltétel).

Definíció. Funkcionális függések egy E halmazának egy minimális lefedése (standard kanonikus alakú) függéseknek egy olyan minimális (redundanciamentes) halmaza, amely ekvivalens E-vel. A 10.2-es algoritmus segítségével bármilyen E függéshalmazhoz előállítható *legalább egy* minimális F lefedés.

Ha több olyan függéshalmaz is létezik, amely a fenti definíció szerint E minimális lefedése, akkor a *minimalitáshoz* további kritériumokat szoktunk megfogalmazni. Választhatjuk például az a minimális halmazt, amely a *legkevesebb függést* tartalmazza, vagy amely a legkisebb *teljes hosszal* rendelkezik (egy függéshalmaz teljes hosszát úgy számítjuk ki, hogy összefűzzük a függéseket, és egy hosszú karaktersorozatnak tekintjük).

2. algoritmus − Funkcionális függések egy *E* halmazához az *F* minimális lefedés meghatározása.

1. Legyen $F := E$.
2. Cseréljük ki az összes F-beli $X \to \{ A_1, A_2, \ldots, A_n \}$ az $X \to A_1, X \to A_2, \ldots, X \to A_n$ n darab funkcionális függéssel.
3. Minden F-beli $X \to A$ funkcionális függésre
   - minden X-beli B attribútumra
     - ha $(F - \{ X \to A \}) \cup \{ (X - \{ B \}) \to A \}$ ekvivalens F-fel,
       - akkor cseréljük ki F-ben $X \to A$-t $(X - \{ B \}) \to A$-ra.
4. Az összes többi F-beli $X \to A$ funkcionális függésre
   - ha az $F - \{ X \to A \}$ ekvivalens F-fel,
     - akkor töröljük F-ből $X \to A$-t.

A fenti algoritmust az alábbi példával illusztráljuk:

Legyen adott funkcionális függések egy $E = \{ B \to A, D \to A, AB \to D \}$ halmaza. Határozzuk meg E egy minimális lefedését!

- ❒ A felsorolt függések mind kanonikus alakúak; így teljesítettük a 2. algoritmus 2. lépését, és mehetünk tovább a 3. lépésre. A 3. lépésben el kell döntenünk, hogy az $AB \to D$ bal oldalán van-e redundáns attribútum; azaz helyettesíthetjük-e $A \to D$-vel vagy $B \to D$-vel.
- ❒ Ha a $B \to A$-t mindkét oldalon bővítjük B-vel (IR2), akkor $BB \to AB$-t, azaz röviden (i) $B \to AB$-t kapjuk. Ezenkívül (ii) $AB \to D$ is adott.
- ❒ Ekkor a tranzitivitás szabályát (IR3) alkalmazva (i)-re és (ii)-re, kapjuk $B \to D$-t. Az $AB \to D$ tehát kicserélhető $B \to D$-re.
- ❒ Van tehát egy, az eredeti E-vel ekvivalens halmazunk, $E' = \{ B \to A, D \to A, B \to D \}$. A

harmadik lépésben nem lehet további redukciót végrehajtani, mivel mindegyik függésnek egyetlen attribútum áll a bal oldalán.

- A 4. lépésben redundáns funkcionális függéseket keresünk E'-ben. A tranzitivitás szabályát alkalmazva $B \rightarrow D$-re és $D \rightarrow A$-ra, $B \rightarrow A$-t kapjuk. $B \rightarrow A$ tehát redundáns E'-ben, így törölhető.
- E minimális lefedése tehát $\{ B \rightarrow D, D \rightarrow A \}$.

A 9. fejezetben látni fogjuk, hogyan szintetizálhatunk relációkat függések egy E halmazából úgy, hogy először meghatározzuk E-nek a minimális lefedését.

Az elsődleges kulcson alapuló normálformák

Most, hogy áttanulmányoztuk a funkcionális függéseket és néhány tulajdonságukat, készen állunk arra, hogy segítségükkel megadjuk a relációsémák szemantikájának néhány aspektusát. Feltételezzük, hogy minden relációhoz adott funkcionális függések egy halmaza, és hogy minden relációnak van egy kijelölt elsődleges kulcsa; ezek az információk — a normálformákra vonatkozó tesztekkel (feltételekkel) együtt — képezik az alapját a relációséma-tervezés *normalizációs folyamatának*. A legtöbb, gyakorlatban alkalmazott relációséma-tervezési projekt az alábbi két megközelítés egyikét alkalmazza:

- Hajtsunk végre koncepcionális sématervezést egy koncepcionális modell (például ER vagy EER) segítségével, majd képezzük le a koncepcionális tervet relációsémák halmazára.
- Tervezzük meg a relációkat olyan külső ismeretek alapján, melyeket már létező állományokból (iratokból), űrlapokból vagy jelentésekből származtatunk.

Bármelyik megközelítést is választjuk, célszerű a relációkat kiértékelni jóság szempontjából, és szükség szerint tovább bontani őket a magasabb normálformák elérése érdekében az ebben és a következő fejezetben bemutatott normalizálási elmélet felhasználásával. Ebben az alfejezetben a relációsémák első három normálformájával és a mögöttük meghúzódó tartalommal, valamint történeti fejlődésükkel foglalkozunk. A . alfejezetben ezen normálformák általánosabb definícióit ismertetjük, amelyek a reláció összes kulcsjelöltjét figyelembe veszik, nem csak az elsődleges kulcsot.

A normálformák informális tárgyalásával és a kialakulásuk mögött rejlő motivációval kezdünk, valamint áttekintünk néhány olyan, a 5. fejezetben ismertetett definíciót, amelyekre itt is szükségünk lesz. Ezután az első normálformát (1NF) ismertetjük a . alfejezetben, majd a második normálforma (2NF) és a harmadik normálforma (3NF) elsődleges kulcson alapuló definícióit mutatjuk be a . és . alfejezetekben.

Relációk normalizálása

A normalizációs folyamat Codd első javaslata szerint (Codd (1972)) végigvisz egy relációsémát tesztek egy sorozatán annak *ellenőrzésére*, hogy kielégít-e egy bizonyos **normálformát**. Ezt a folyamatot, amely felülről lefelé haladva minden relációt kiértékel a normálformákra vonatkozó kritériumok szerint, és ha szükséges, fölbontja őket, *analízis alapú relációs tervezésnek* tekinthetjük. Codd kezdetben három normálformát javasolt, amelyeket első, második és harmadik normálformának nevezett. Boyce és Codd később egy, a 3NF-nél erősebb normálformát definiált, amelyet Boyce–Codd-normálformának (BCNF) nevezünk. Mindegyik normálforma egyetlen analitikai eszközön alapul: a reláció attribútumai között fennálló funkcionális függéseken. Még később bevezetésre került a negyedik (4NF) és az ötödik normálforma (5NF), amelyek a többértékű, illetve a kapcsolásfüggés fogalmán alapulnak; ezekről a 9. fejezetben lesz szó. A 9. fejezet elején azt is megmutatjuk, hogyan lehet 3NF relációkat összerakni funkcionális függések adott halmazából. Ezt a megközelítést *szintézis alapú relációs tervezésnek* nevezzük.

Az adatok normalizálása egy olyan folyamatnak tekinthető, amely elemzi az adott relációsémákat a funkcionális függéseik és elsődleges kulcsaik alapján, hogy (1) minimalizálja a redundanciát és (2) minimalizálja a . alfejezetben tárgyalt beszúrási, törlési és módosítási anomáliákat. A nem megfelelő relációsémákat, amelyek nem felelnek meg bizonyos feltételeknek — **normálformateszteknek** — felbontjuk kisebb relációsémákra, amelyek megfelelnek a teszteknek, és így rendelkeznek az elvárt tulajdonságokkal. A normalizációs eljárás tehát az adatbázistervezők számára a következőket biztosítja:

- Egy formális keretrendszert a relációsémák elemzéséhez, amely a relációsémák kulcsain és az attribútumaik között fennálló funkcionális függéseken alapszik.
- Normálformatesztek egy sorozatát, amelyek egy-egy relációsémán hajthatók végre, hogy a relációs adatbázis tetszőleges mértékig normalizálható legyen.

Definíció. Egy reláció **normálformája** a legmagasabb olyan normálforma, amely feltételének a reláció megfelel, és így jelzi a reláció normalizáltságának a fokát.

A normálformák önmagukban, más tényezőktől *elkülönítve*, nem garantálnak jó adatbázistervet. Általában nem elegendő külön ellenőrizni, hogy az egyes relációsémák az adatbázisban mondjuk BCNF-ben vagy 3NF-ben vannak-e. Ehelyett a dekompozíción alapuló normalizálási folyamatnak olyan további tulajdonságok meglétét is biztosítania kell, amelyek a relációsémák együttesére kell, hogy teljesüljenek. Két ilyen tulajdonság van:

- A **veszteségmentes join** vagy **nemadditív join tulajdonság**, amely garantálja, hogy a . alfejezetben ismertetett álrekordgenerálási probléma nem lép fel a felbontás során keletkező relációsémák tekintetében.
- A **függésmegőrző tulajdonság**, amely biztosítja, hogy minden funkcionális függés megmarad a felbontást követően kapott valamelyik relációban.

A nemadditív join tulajdonság különösen fontos és **bármi áron biztosítani kell**, míg a függésmegőrző tulajdonságot, bár kívánatos, néha feláldozzuk, ahogy azt a . alfejezetben majd látni fogjuk. A fenti két tulajdonságot garantáló formális fogalmakat és technikákat a 9. fejezetben fogjuk ismertetni.

Normálformák gyakorlati alkalmazása

A legtöbb gyakorlati tervező projekt adatbázisok létező terveiből indul ki, amelyek korábbi tervekből, korai modellek alapján készített tervekből vagy létező állományokból származnak. A gyakorlatban a normalizálást úgy hajtják végre, hogy az eredményül kapott tervek jó minőségűek, és megfelelnek az előzőekben ismertetett elvárt tulajdonságoknak. Bár számos magasabb normálformát definiáltak (mint például a 4NF és az 5NF, amelyeket a 9. fejezetben ismertetünk), ezek gyakorlati hasznossága kérdésessé válik akkor, amikor azok az adatbázistervezők és felhasználók, akiknek ez a feladatuk, nehezen tudják megérteni, illetve felismerni azokat a megszorításokat, amelyeken ezek a normálformák alapulnak. Az iparban manapság alkalmazott adatbázistervezés során ezért csak a 3NF-ig, BCNF-ig vagy 4NF-ig végrehajtott normalizálásra fordítanak figyelmet.

Megemlíthetjük még, hogy az adatbázistervezőknek *nem szükséges* a lehető legmagasabb fokig normalizálniuk. A relációkat alacsonyabb normalizációs szinten is hagyhatjuk (például 2NF-ben) a teljesítmény növelése érdekében, ahogy arról a . alfejezet végén írtunk.

Definíció. Azt a folyamatot, amelynek során magasabb normálformájú relációk összekapcsolását alaprelációként tároljuk (amely alacsonyabb normálformában van), **denormalizációnak** nevezzük.

Kulcsok és a kulcsokat alkotó attribútumok definíciói

Mielőtt tovább haladnánk, vessünk ismételten egy pillantást egy relációséma kulcsainak a definícióira, ahogyan azt a 5. fejezetben tettük.

Definíció. Egy $R = \{A_1, A_2, \ldots, A_n\}$ relációséma **szuperkulcsa** azon $S \subseteq R$ attribútumhalmaz, amelyre teljesül, hogy bármely R feletti legális r relációban nincs két olyan t_1 és t_2 rekord, amelyre $t_1[S] = t_2[S]$ teljesül. A K **kulcs** egy olyan szuperkulcs, amelyből bármely attribútum eltávolítása azt eredményezi, hogy K már nem lesz szuperkulcs.

A különbség egy kulcs és egy szuperkulcs között az, hogy a kulcsnak *minimálisnak* kell lennie, azaz ha van egy $K = \{A_1, A_2, \ldots, A_k\}$ kulcsa R-nek, akkor $K - \{A_i\}$ nem kulcsa R-nek egyetlen A_i ($1 \leq i \leq k$) esetén sem. A 8.1. ábrán {Szsz} egy kulcsa a DOLGOZÓ-nak, míg az {Szsz}, {Szsz, Dnév}, {Szsz, Dnév, Szdátum} és minden olyan attribútumhalmaz, amely tartalmazza Szsz-t, szuperkulcs.

Ha egy relációsémának egynél több kulcsa van, akkor ezeket **kulcsjelölteknek** nevezzük. A kulcsjelöltek közül egy *tetszőlegesen* kiválasztott lesz az **elsődleges kulcs**, a többit másodlagos kulcsoknak nevezzük. Minden relációsémának kell, hogy legyen elsődleges kulcsa. A 8.1. ábrán {Szsz} az egyetlen kulcsjelöltje a DOLGOZÓ-nak, így egyben ez az elsődleges kulcs is.

Definíció. Az R relációséma egy attribútumát R egy **elsődleges attribútumának** nevezzük, ha eleme R *valamely kulcsjelöltjének*. Egy attribútumot **másodlagos** vagy **leíró** attribútumnak hívunk, ha nem elsődleges attribútum, azaz nem eleme egyetlen kulcsjelöltnek sem.

A 8.1. ábrán az Szsz is és a Pszám is elsődleges attribútumai a DOLGOZIK_RAJTA relációsémának, míg a DOLGOZIK_RAJTA többi attribútuma leíró attribútum.

Mindezek után most bemutatjuk az első három normálformát: az 1NF-et, a 2NF-et és a 3NF-et. Ezeket Codd ajánlotta (Codd (1972)) mint egy olyan sorozatot, amellyel elérhető a relációk kívánatos 3NF állapota, áthaladva az 1NF és 2NF közbenső állapotain, ha szükséges. Amint azt látni fogjuk, a 2NF és a 3NF különböző problémákat orvosolnak. Mindazonáltal történeti okokból kifolyólag szokás ebben a sorrendben ismertetni őket, ezért feltételezni fogjuk, hogy egy 3NF reláció *már kielégíti* a 2NF-et.

Első normálforma

Az **első normálforma** az alap relációs modellben ismertetett reláció fogalom formális definíciójának a részeként is felfogható;[32] történelmileg azért definiálták, hogy megtiltsa a többértékű attribútumokat, az összetett attribútumokat és ezek kombinációját. Azt mondja ki, hogy az attribútumok tartománya kizárólag *atomi* (egyszerű, oszthatatlan) *értékeket* tartalmazhat, és hogy a rekordokban bármely attribútum értéke csak *egyetlen érték* lehet az adott attribútum tartományából. Az 1NF ezáltal megtiltja, hogy *egy rekordon belül* az attribútumok értéke egy értékhalmaz, egy érték n-es vagy ezek kombinációja legyen. Más szóval, az 1NF nem engedi meg *a relációkon belüli relációkat*, illetve *a relációkat mint attribútumértékeket a rekordokon belül*. Az 1NF szerint tehát egy attribútum értéke kizárólag egyetlen **atomi** (vagy **oszthatatlan**) **érték** lehet.

8.8. ábra - Normalizálás 1NF-be. (a) Egy relációséma, amely nincs 1NF-ben. (b) Az OSZTÁLY reláció példa állapota. (c) Ugyanazon reláció 1NF változata redundanciával.

(a)

OSZTÁLY

Onév	Oszám	Ovez_szsz	Ohelyszínek

(b)

OSZTÁLY

Onév	Oszám	Ovez_szsz	Ohelyszínek
Kutatás	5	2 551208 2219	{ Vác, Tiszafüred, Budapest }
Humán erőforrás	4	2 690329 1099	{ Kecskemét }
Központ	1	1 371110 4518	{ Budapest }

(c)

OSZTÁLY

Onév	Oszám	Ovez_szsz	Ohelyszín
Kutatás	5	2 551208 2219	Vác
Kutatás	5	2 551208 2219	Tiszafüred
Kutatás	5	2 551208 2219	Budapest
Humán erőforrás	4	2 690329 1099	Kecskemét
Központ	1	1 371110 4518	Budapest

Tekintsük a 8.1. ábrán látható OSZTÁLY relációsémát, amelynek az elsődleges kulcsa az Oszám, és tegyük fel, hogy kibővítjük egy Ohelyszínek attribútummal, ahogyan a 8.8. (a) ábrán látható. Feltételezzük, hogy minden osztály *több* helyszínnel is rendelkezhet. Az OSZTÁLY séma és egy példa relációállapot látható a 8.8. ábrán. Ahogy látható, nincs 1NF-ben, mert az Ohelyszínek nem atomi attribútum, ahogy a 8.8. (b) ábra első rekordja mutatja. Kétféleképpen tekinthetünk az Ohelyszínek attribútumra:

- Az Ohelyszínek tartománya atomi értékekből áll ugyan, de egyes rekordok ezen értékek halmazát tartalmazhatják. Ebben az esetben az Ohelyszínek nem függ funkcionálisan az Oszám elsődleges kulcstól.
- Az Ohelyszínek tartománya értékhalmazokból áll, ezért nem atomi. Ebben az esetben fennáll az Oszám → Ohelyszínek függés, mert minden ilyen értékhalmaz az attribútum tartományának egyetlen elemeként tekinthető.[33]

A 8.8. ábrán látható OSZTÁLY reláció egyik esetben sincs 1NF-ben; valójában még relációnak sem tekinthető az . alfejezetben definiált reláció fogalom alapján. Három fő technika létezik az első normálforma elérésére az ilyen relációk esetén.

1. Távolítsuk el az 1NF-et sértő Ohelyszínek attribútumot, és helyezzük el egy külön OSZT_HELYSZÍNEK relációban az OSZTÁLY Oszám elsődleges kulcsával együtt. Az új reláció elsődleges kulcsa az {Oszám, Ohelyszín} lesz, ahogy a 8.2. ábrán látható. Az OSZT_HELYSZÍNEK relációban az egyes osztályok *minden helyszínéhez* külön rekord tartozik. Ezzel felbontottuk a nem 1NF relációt két 1NF relációra.
2. Bővítsük ki a kulcsot az eredeti OSZTÁLY relációban úgy, hogy külön rekord tartozzon az osztályok minden egyes helyszínéhez, ahogy a 8.8. (c) ábrán látható. Ebben az esetben az elsődleges kulcs az {Oszám, Ohelyszín} lesz. A megoldás hátránya, hogy *redundanciát* vezet be a relációba.
3. Ha tudjuk, hogy az attribútum egy *maximális számú értéket vehet fel* — például tudjuk, hogy *legfeljebb három helyszín* tartozhat egy osztályhoz —, akkor helyettesítsük az Ohelyszínek

attribútumot három atomi attribútummal: Ohelyszín1, Ohelyszín2 és Ohelyszín3. Ennek a megoldásnak a hátránya, hogy *NULL értékeket* vezet be, ha a legtöbb osztálynak háromnál kevesebb helyszíne van. Ráadásul hamis értelmezést sugall a helyszínek rendezettségére vonatkozóan, pedig ilyen rendezettség eredetileg nem is szerepelt a terveink között. Nehezebbé válik a lekérdezés is ezen attribútum alapján; például képzeljük el, hogyan írnánk meg az alábbi lekérdezést egy ilyen sémát használva: *Listázzuk ki azokat az osztályokat, amelyiknek az egyik helyszíne Vác!*

A fent említett három megoldás közül általában az elsőt tekintjük a legjobbnak, mert nem okoz redundanciát, és teljesen általános, nincs korlát az értékek maximális számára vonatkozóan. Valójában ha a második megoldást választjuk, akkor a későbbi normalizációs lépésekben további dekompozícióval úgyis az első megoldáshoz jutunk.

8.9. ábra - Beágyazott relációk normalizálása 1NF-be. (a) A DOLG_PROJ reláció sémája a Projektek *beágyazott reláció* attribútummal. (b) A DOLG_PROJ reláció példa állapota mutatja a beágyazott relációkat mindegyik rekordban. (c) A DOLG_PROJ szétbontása DOLG_PROJ1 és DOLG_PROJ2 relációkra az elsődleges kulcs hozzávételével.

(a)

DOLG_PROJ

		Projektek	
Szsz	Dnév	Pszám	Órák

(b)

DOLG_PROJ

Szsz	Dnév	Pszám	Órák
1 650109 0812	Kovács László	1	32.5
		2	7.5
2 620915 3134	Horváth Erzsébet	3	40.0
1 720731 2985	Tóth János	1	20.0
		2	20.0
2 551208 2219	Szabó Mária	2	10.0
		3	10.0
		10	10.0
		20	10.0
1 680119 6749	Kiss István	30	30.0
		10	10.0
2 690329 1099	Fazekas Ilona	10	35.0
		30	5.0
1 410620 4902	Takács József	30	20.0
		20	15.0
1 371110 4519	Nagy Zoltán	20	NULL

(c)

DOLG_PROJ1

Szsz	Dnév

DOLG_PROJ2

Szsz	Pszám	Órák

Az első normálforma az olyan többértékű attribútumokat is tiltja, amelyek maguk is összetettek. Ezeket **beágyazott relációknak** hívjuk, mert minden rekord egy relációt *foglalhat magában*. A 8.9.

ábra mutatja, hogy hogy nézne ki a DOLG_PROJ reláció, ha megengednénk a beágyazást. Minden rekord egy dolgozó egyedet reprezentál, és *az egyes rekordokon belül* a PROJEKTEK(Pszám, Órák) reláció reprezentálja a dolgozó projektjeit, és hogy hány órát dolgozik az adott dolgozó az egyes projekteken. A DOLG_PROJ reláció sémája a következőképpen írható fel:

- DOLG_PROJ(Szsz, Dnév, {PROJEKTEK(Pszám, Órák)})

A halmazt jelölő kapcsos zárójelek mutatják azt, hogy a PROJEKTEK attribútum többértékű, míg a PROJEKTEK-et alkotó attribútumokat kerek zárójelek között soroljuk fel. Érdekes módon az összetett objektumokat és az XML adatokat támogató mai trendek megkísérlik lehetővé tenni és formalizálni a beágyazott relációk használatát a relációs adatbázisrendszereken belül, amit korábban az 1NF tiltott.

Figyeljük meg, hogy az Szsz a DOLG_PROJ reláció elsődleges kulcsa a 8.9. (a) és (b) ábrákon, míg a Pszám **részleges** kulcsa a beágyazott relációnak; ez azt jelenti, hogy a Pszámnak a beágyazott relációban minden egyes rekordon belül egyedi értékkel kell rendelkeznie. Ahhoz, hogy 1NF-be normalizáljuk ezt a relációt, kivesszük azt az attribútumot, amely beágyazott relációként szerepel, új relációt készítünk belőle, és *hozzávesszük az eredeti reláció elsődleges kulcsát*; az új reláció elsődleges kulcsa az eredeti reláció elsődleges kulcsának és a részleges kulcsnak a kombinációja lesz. A felbontás és az elsődleges kulcs átvétele eredményezi a DOLG_PROJ1 és DOLG_PROJ2 sémákat, ahogyan a 8.9. (c) ábrán látható.

Ezt az eljárást rekurzívan alkalmazhatjuk egy olyan relációra, amely többszintű beágyazást tartalmaz, hogy **eltávolítsuk a beágyazásokat** a relációból, és 1NF relációkat hozzunk létre. Ez akkor hasznos, ha egy nem normalizált, többszintű beágyazást tartalmazó relációsémát szeretnénk átkonvertálni 1NF relációkba. Óvatosan kell kezelni azt a szituációt, amikor egynél több többértékű attribútum fordul elő egy relációban. Tekintsük például a következő nem 1NF relációt:

- SZEMÉLY(Szsz, {Rendszám}, {Telefonszám})

Ez a reláció azt a tényt szemlélteti, hogy egy személynek több autója és több telefonja lehet. Ha a fenti második lehetőséghez hasonló stratégiát követünk, akkor egy olyan relációt kapunk, amelynek kulcsát az összes attribútum együtt alkotja:

- SZEMÉLY_1NF(Szsz, Rendszám, Telefonszám)

Hogy elkerüljük a Rendszám és a Telefonszám közötti extra, nem létező kapcsolatok bevezetését, az értékeik minden lehetséges kombinációját szerepeltetjük minden Szsz esetén, megnövelve ezzel a redundanciát. Ez olyan problémákhoz vezet, amelyeket többértékű függésekkel és a 4NF-fel kezelünk, amelyekről a 9. fejezetben lesz szó. A fentebb bemutatott SZEMÉLY relációsémában lévő két többértékű attribútum kezelésének helyes módja, hogy szétbontjuk a relációsémát két külön relációra a korábban tárgyalt első stratégiát követve: P1(Szsz, Rendszám) és P2(Szsz, Telefonszám).

Második normálforma

A **második normálforma (2NF)** a *teljes funkcionális függés* fogalmán alapul. Egy $X \rightarrow Y$ funkcionális függés **teljes funkcionális függés**, ha az X-ből egy A attribútum eltávolításával a függés már nem áll fenn; azaz bármely $A \in X$ attribútum esetén $(X - \{A\})$ *nem* határozza meg funkcionálisan Y-t. Egy $X \rightarrow Y$ funkcionális függés **részleges függés**, ha valamely $A \in X$ attribútum eltávolítható X-ből úgy, hogy a függés továbbra is fennáll; azaz valamely $A \in X$ esetén $(X - \{A\}) \rightarrow Y$. A 8.3. (b) ábrán az { Szsz, Pszám } → Órák teljes függés (sem Szsz → Órák, sem Pszám → Órák nem áll fenn). Az { Szsz, Pszám } → Dnév ellenben részleges függés, mert Szsz → Dnév teljesül.

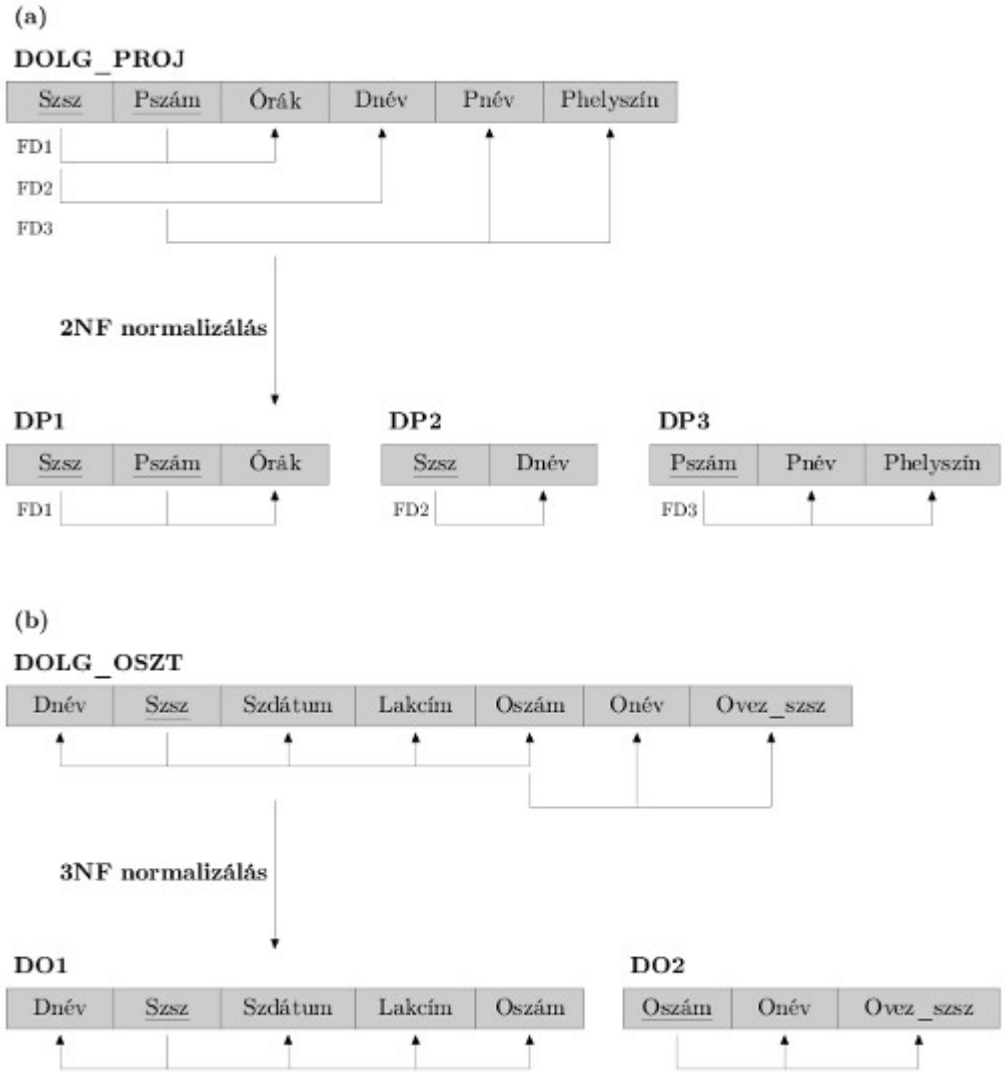
Definíció. Az R relációséma 2NF-ben van, ha R minden A másodlagos (leíró) attribútuma *teljesen funkcionálisan függ* R elsődleges kulcsától.

A 2NF tesztelése az olyan funkcionális függések tesztelését jelenti, amelyeknek a bal oldali attribútumai az elsődleges kulcs részei. Ha az elsődleges kulcs egyetlen attribútumot tartalmaz,

akkor a tesztet egyáltalán nem kell alkalmazni. A 8.3. (b) ábrán látható DOLG_PROJ reláció 1NF-ben van, de nincs 2NF-ben. A Dnév másodlagos attribútum sérti a 2NF-et az FD2 miatt, mint ahogy a Pnév és a Phelyszín másodlagos attribútumok is az FD3 miatt. Az FD2 és FD3 funkcionális függések a Dnév, Pnév és Phelyszín attribútumokat részlegesen függővé teszik a DOLG_PROJ {Szsz, Pszám} elsődleges kulcsától, megsértve ezzel a 2NF tesztet.

Ha egy relációséma nincs 2NF-ben, akkor *második normálformára* vagy *2NF-re hozhatjuk*, azaz felbonthatjuk több 2NF relációsémára, amelyekben a másodlagos attribútumok az eredeti elsődleges kulcsnak csak azon részével szerepelnek együtt, amelyektől teljesen funkcionálisan függnek. Ezek alapján a 8.3. (b) ábrán látható FD1, FD2 és FD3 funkcionális függések a DOLG_PROJ három új (DP1, DP2, DP3) relációsémára történő felbontásához vezetnek, amelyek mindegyike 2NF-ben van, ahogyan az a 8.10. (a) ábrán látható.

8.10. ábra - Normalizálás 2NF-be és 3NF-be. (a) A DOLG_PROJ normalizálása 2NF relációkra. (b) A DOLG_OSZT normalizálása 3NF relációkra.



Harmadik normálforma

A **harmadik normálforma (3NF)** a *tranzitív függés* fogalmán alapul. Egy R relációséma $X \rightarrow Y$ funkcionális függése **tranzitív függés**, ha létezik egy olyan Z attribútumhalmaz, amely nem kulcsjelölt és nem része R egyetlen kulcsának sem[34], és fennáll $X \rightarrow Z$, illetve $Z \rightarrow Y$. A 8.3. (a) ábra DOLG_OSZT relációsémájában az Szsz → Ovez_szsz tranzitív függés az Oszámon keresztül, mert az Szsz → Oszám és az Oszám → Ovez_szsz is fennáll, *és* az Oszám nem kulcs és nem is

része a DOLG_OSZT kulcsának. Intuitívan láthatjuk, hogy az Ovez_szsz függése az Számtól nem kívánatos a DOLG_OSZT relációsémában, mivel az Szám nem kulcsa a DOLG_OSZT-nak.

Definíció. Codd eredeti definíciója alapján egy *R* relációséma **harmadik normálformában (3NF-ben)** van, ha 2NF-ben van *és R* egyik másodlagos (leíró) attribútuma sem függ tranzitívan az elsődleges kulcstól.

A 8.3. (a) ábrán a DOLG_OSZT relációséma 2NF-ben van, mivel nincsen benne kulcstól való részleges függés. Nincs azonban 3NF-ben, mert az Ovez_szsz (és az Onév is) tranzitívan függ az Szsz-től az Oszámon keresztül. A DOLG_OSZT relációsémát úgy normalizálhatjuk, hogy felbontjuk két 3NF relációsémára (DO1 és DO2), ahogy a 8.10. (b) ábrán látható. Intuitívan láthatjuk, hogy az DO1 és a DO2 a dolgozókról és az osztályokról egymástól független tényeket tartalmaznak. A DO1-en és DO2-n végrehajtott természetes összekapcsolás álrekordok előállítása nélkül visszaadja az eredeti DOLG_OSZT relációt.

Intuitívan láthatjuk, hogy minden olyan funkcionális függés, amelynek a bal oldala része (valódi részhalmaza) az elsődleges kulcsnak, illetve minden olyan funkcionális függés, amelynek a bal oldala egy nem kulcs attribútum, az *problematikus* funkcionális függés. A 2NF és 3NF normalizálás megszünteti ezeket a problémás funkcionális függéseket azáltal, hogy az eredeti relációt új relációkra bontja szét. A normalizálási folyamat szempontjából nem szükséges a részleges függéseket a tranzitív függések előtt eltüntetni, de történelmileg a 3NF-et azzal a feltételezéssel definiálták, hogy a relációt már teszteltük a 2NF-re, mielőtt a 3NF-re tesztelnénk. A 8.1. táblázat informálisan összefoglalja a három, elsődleges kulcson alapuló normálformát, az egyes esetekben használt teszteket és a megfelelő megoldást, azaz a kívánt normálforma eléréséhez szükséges tennivalókat.

8.1. táblázat - Az elsődleges kulcsra épülő normálformák és az elérésükhöz szükséges normalizálási tevékenységek összefoglalása

Normálforma	Tesztelés	Megoldás (normalizáció)
Első (1NF)	A relációnak nincs többértékű attribútuma, és nincs benne beágyazott reláció.	Képezzünk új relációt minden egyes többértékű attribútumhoz és beágyazott relációhoz.
Második (2NF)	Azon relációkban, amelyeknek az elsődleges kulcsa több attribútumból áll, egyetlen leíró attribútum sem függ az elsődleges kulcs egy részétől.	Bontsuk szét a relációt, és hozzunk létre új relációt minden egyes részleges kulcshoz az általuk meghatározott attribútumokkal. Győződjünk meg róla, hogy megtartunk egy relációt az eredeti elsődleges kulccsal és azokkal az attribútumokkal, amelyek teljesen függnek tőle.
Harmadik (3NF)	A relációnak nincs olyan leíró attribútuma, amely funkcionálisan függ egy másik leíró attribútumtól (vagy leíró attribútumok egy halmazától). Azaz egyetlen leíró attribútum sem függ tranzitívan az elsődleges kulcstól.	Bontsuk szét a relációt, és hozzunk létre egy új relációt, amely tartalmazza az(oka)t a leíró attribútumo(ka)t, amely(ek) funkcionálisan meghatároz(nak) más leíró attribútumo(ka)t.

A második és harmadik normálforma általános definíciója

Általában úgy szeretnénk megtervezni relációsémáinkat, hogy se részleges függéseket, se tranzitív függéseket ne tartalmazzanak, mivel az ilyen típusú függések okozzák a . alfejezetben tárgyalt karbantartási anomáliákat. A 3NF alakú relációkra történő normalizálási lépések, amelyeket eddig

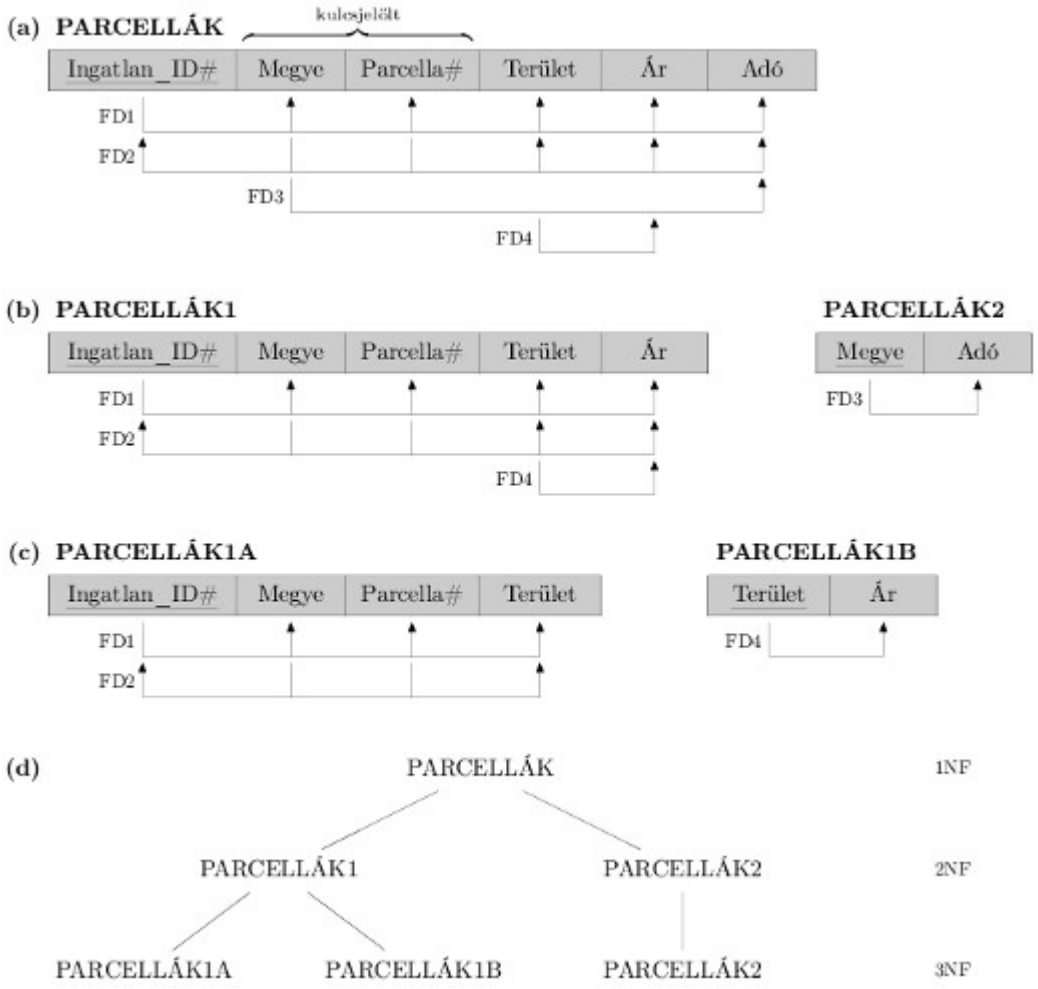
tárgyaltunk, el is távolították az *elsődleges kulcstól* való részleges és tranzitív függéseket. Ezek a definíciók azonban nem veszik figyelembe a reláció többi kulcsjelöltjét, már ha léteznek ilyenek egyáltalán. Ebben a fejezetben megadjuk a 2NF és a 3NF általánosabb definícióit, amelyek már a reláció *összes* kulcsjelöltjét figyelembe veszik. Megjegyzendő, hogy ez nincs hatással az 1NF definíciójára, mivel az független a kulcsoktól és a funkcionális függésektől. Az **elsődleges attribútum** általános definíciójaként azt az attribútumot fogjuk elsődlegesnek tekinteni, amelyik eleme *bármely kulcsjelöltnek*. A részleges és a teljes funkcionális függéseket, valamint a tranzitív függést a reláció *összes kulcsjelöltjére vonatkoztatva* fogjuk figyelembe venni.

A második normálforma általános definíciója

Definíció. Egy *R* relációséma **második normálformában** (2NF-ben) van, ha egyetlen *R*-beli másodlagos (leíró) *A* attribútum sem függ részlegesen *R egyetlen* kulcsától sem.

A 2NF ellenőrzése magában foglalja mindazoknak a funkcionális függéseknek az ellenőrzését, amelyeknek a bal oldali attribútumai az elsődleges kulcs *részei*. Ha az elsődleges kulcs egyetlen attribútumot tartalmaz, akkor az ellenőrzést egyáltalán nem kell elvégezni. Tekintsük a 8.11. (a) ábrán bemutatott PARCELLÁK relációsémát, amely az ország különböző megyéiben eladásra kínált földterületeket írja le. Tegyük föl, hogy két kulcsjelölt van: az Ingatlan_ID# és a {Megye, Parcella#}; azaz a parcellaszámok csak egy-egy megyén belül egyediek, míg az Ingatlan_ID# számok az egész országra vonatkozóan azok.

8.11. ábra - Normalizálás 2NF-be és 3NF-be. (a) A PARCELLÁK reláció négy funkcionális függéssel (FD1−FD4). (b) A szétbontott, második normálformájú PARCELLÁK1 és PARCELLÁK2 relációk. (c) A PARCELLÁK1 szétbontása a harmadik normálformájú PARCELLÁK1A és PARCELLÁK1B relációkra. (d) A PARCELLÁK normalizálási folyamatának az összefoglalása.



Az Ingatlan_ID# és a {Megye, Parcella#} kulcsjelölteket figyelembe véve tudjuk, hogy a 8.11. (a) ábra FD1 és FD2 funkcionális függései fennállnak. Az Ingatlan_ID#-t választjuk elsődleges kulcsnak, amit aláhúzással jelölünk a 8.11. (a) ábrán, bár semmilyen különösebb tényező nem szól emellett a kulcs mellett a másik kulcsjelölttel szemben. Tegyük fel, hogy a következő két további funkcionális függés is fennáll a PARCELLÁK sémában:

FD3: Megye → Adó

FD4: Terület → Ár

Más szóval, az FD3 függés azt mondja, hogy az adó mértéke rögzített egy adott megyében (nem változik parcelláról parcellára ugyanazon megyén belül), míg az FD4 azt mondja, hogy egy parcella árát meghatározza a terület nagysága, függetlenül attól, hogy melyik megyében is található. (Feltételezve, hogy ez a parcellának az adózáskor figyelembe vett ára.)

A PARCELLÁK relációséma megsérti a 2NF általános definícióját, mert az Adó részlegesen függ a {Megye, Parcella#} kulcsjelölttől, az FD3 miatt. A PARCELLÁK 2NF-re történő normalizálásához felbontjuk a sémát két relációra, a 8.11. (b) ábrán bemutatott PARCELLÁK1-re és PARCELLÁK2-re. A 2NF-et megsértő Adó attribútumnak a PARCELLÁK-ból történő eltávolításával létrehozzuk a PARCELLÁK1-et, az Adót pedig a Megyével (a részleges függést okozó FD3 bal oldalával) együtt a másik, PARCELLÁK2 relációba tesszük. Ezzel mind a PARCELLÁK1, mind a PARCELLÁK2 2NF-ben lesz. Megjegyzendő, hogy az FD4 nem sértette meg a 2NF-et, és továbbra is érvényes maradt a PARCELLÁK1-ben.

A harmadik normálforma általános definíciója

Definíció. Egy *R* relációséma **harmadik normálformában** (3NF-ben) van, ha valahányszor egy *X*

→ *A nemtriviális* funkcionális függés fennáll *R*-en, akkor vagy (a) *X* szuperkulcsa *R*-nek, vagy (b) *A* elsődleges attribútuma *R*-nek.

Ennek a definíciónak megfelelően a PARCELLÁK2 (a 8.11. (b) ábrán) 3NF-ben van. Ugyanakkor az FD4 a PARCELLÁK1-ben megsérti a 3NF-et, mert a Terület nem szuperkulcs, és az Ár sem elsődleges attribútum a PARCELLÁK1-ben. Ahhoz, hogy a PARCELLÁK1-et 3NF-re hozzuk, fel kell bontanunk őt a 8.11. (c) ábrán látható PARCELLÁK1A és PARCELLÁK1B relációsémákra. A 3NF-et megsértő Ár attribútumnak a PARCELLÁK1-ből történő eltávolításával létrehozzuk a PARCELLÁK1A-t, az Árat pedig a Területtel (a tranzitív függést okozó FD4 bal oldalával) együtt a másik, PARCELLÁK1B relációba tesszük. Ezzel mind a PARCELLÁK1A, mind a PARCELLÁK1B 3NF-ben lesz.

Két dolgot érdemes megjegyezni ezzel a példával és a 3NF általános definíciójával kapcsolatban:

- A PARCELLÁK1 megsérti a 3NF-et, mert az Ár tranzitívan függ a PARCELLÁK1 kulcsjelöltjeinek mindegyikétől a másodlagos Terület attribútumon keresztül.
- Ez az általános definíció *közvetlenül* alkalmazható annak eldöntésére, hogy egy relációséma 3NF-ben van-e; *nem* kell megvizsgálnunk előbb a 2NF teljesülését. Ha a 3NF fenti definícióját alkalmazzuk a FD1−FD4 függéseket tartalmazó PARCELLÁK relációsémára, azt találhatjuk, hogy *mind* az FD3, *mind* az FD4 megsértik a 3NF-et. Ezért a PARCELLÁK sémát közvetlenül is felbonthatjuk PARCELLÁK1A-ra, PARCELLÁK1B-re és PARCELLÁK2-re. Így *tetszőleges sorrendben* távolíthatók el azok a tranzitív és részleges függések, amelyek megsértik a 3NF-et.

A harmadik normálforma általános definíciójának értelmezése

Egy *R* relációséma megsérti a 3NF definícióját, ha egy *R*-beli $X \rightarrow A$ funkcionális függés megsérti a 3NF-nek *mind* az (a), *mind* pedig a (b) feltételét. A (b) megsértése azt jelenti, hogy *A* másodlagos (leíró) attribútum. Az (a) megsértése azt jelenti, hogy *X*-nek nem részhalmaza *R* egyetlen kulcsa sem; így *X* másodlagos (leíró), vagy valódi részhalmaza *R* egy kulcsának. Ha *X* nem elsődleges, akkor tipikusan egy tranzitív függésünk van, amely megsérti a 3NF-et, míg ha *X* egy valódi részhalmaza *R* valamely kulcsának, akkor egy részleges függésünk van, amely megsérti a 3NF-et (és a 2NF-et is). Ennélfogva a következőképpen fogalmazhatjuk meg a **3NF egy alternatív általános definícióját**:

Alternatív definíció. Egy *R* relációséma 3NF-ben van, ha *R* minden másodlagos (leíró) attribútumára teljesül a következő két feltétel:

- Az attribútum teljesen funkcionálisan függ *R* minden kulcsától.
- Az attribútum nemtranzitívan függ *R* minden kulcsától.

A Boyce–Codd-féle normálforma

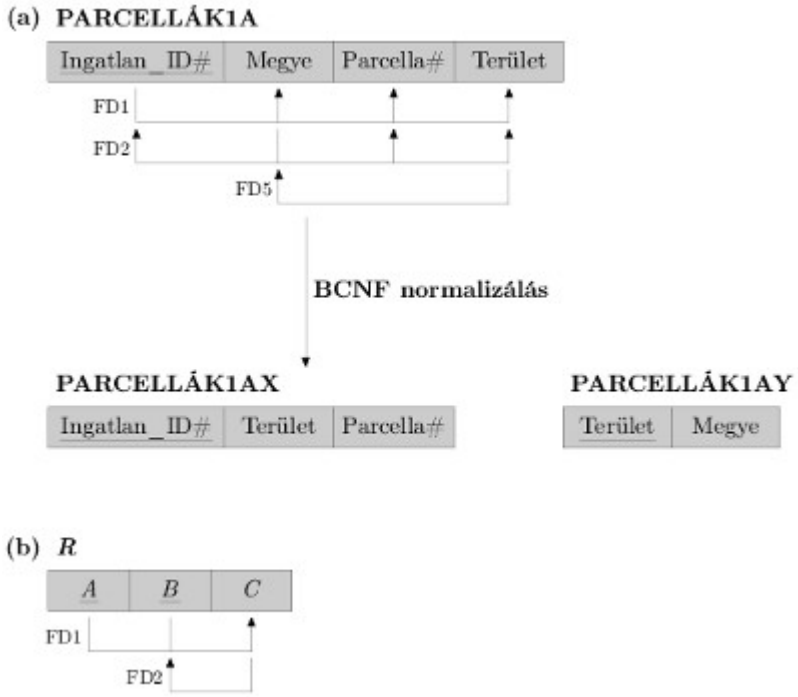
A **Boyce–Codd-féle normálforma** látszólag a 3NF egy egyszerűbb alakja, de valójában erősebb, mint a 3NF. Azaz minden BCNF-ben lévő reláció egyúttal 3NF-ben is van, ám egy 3NF-ben lévő reláció *nem szükségképpen* van BCNF-ben. Intuitívan beláthatjuk a szükségességét egy olyan normálformának, ami erősebb a 3NF-nél, ha visszagondolunk a 8.11. (a) ábra PARCELLÁK relációsémájára a maga 4 funkcionális függésével. Tételezzük föl, hogy parcellák ezrei szerepelnek a relációban, de mindössze csak két megyéből: Baranyából és Csongrádból. Tételezzük föl még továbbá azt is, hogy a parcellaméretek Baranyában csak 0,5, 0,6, 0,7, 0,8, 0,9 és 1,0 hektárosak, míg Csongrádban kizárólag 1,1, 1,2, …, 1,9 és 2,0 hektárosak. Egy ilyen helyzetben felvehetnénk egy újabb funkcionális függést, FD5-öt: Terület → Megye. Ha hozzáadjuk ezt a többi függéshez, a PARCELLÁK1A relációséma még mindig 3NF-ben marad, mert a Megye elsődleges attribútum.

Egy parcella területe, amely meghatározza a megyét, ahogyan azt az FD5 leírja, reprezentálható egy 16 rekordot tartalmazó különálló *R*(Terület, Megye) relációval, mivel csak 16 lehetséges Terület érték létezik. Ez a reprezentáció lecsökkenti a PARCELLÁK1A rekordjainak ezreiben ismétlődő azonos információk redundanciáját. A BCNF egy *erősebb normálforma*, amely nem engedi meg a PARCELLÁK1A-t, hanem arra késztet, hogy felbontsuk azt.

Definíció. Egy *R* relációséma **Boyce–Codd-féle normálformában** (BCNF-ben) van, ha valahányszor egy $X \rightarrow A$ *nemtriviális* funkcionális függés fennáll *R*-ben, akkor *X* egy szuperkulcsa *R*-nek.

A BCNF formális definíciója alig különbözik a 3NF definíciójától. Ez egyetlen különbség a BCNF és a 3NF definíciói között az, hogy a 3NF (b) feltétele, amely megengedi, hogy *A* elsődleges legyen, hiányzik a BCNF-ből. A példánkban FD5 megsérti a BCNF-et a PARCELLÁK1A-ban, mert a Terület nem szuperkulcsa a PARCELLÁK1A-nak. Figyeljük meg, hogy az FD5 kielégíti a 3NF-et a PARCELLÁK1A-ban, mert a Megye elsődleges attribútum ((b) feltétel), de ez a feltétel nem szerepel a BCNF definíciójában. A PARCELLÁK1A-t felbonthatjuk a 8.12. ábrán bemutatott két BCNF-relációra, PARCELLÁK1AX-re és PARCELLÁK1AY-ra. Ez a felbontás elveszíti az FD2 funkcionális függést, mert ennek a függésnek az attribútumai a felbontást követően már nem szerepelnek együtt ugyanazon relációban.

8.12. ábra - Boyce–Codd-féle normálforma. (a) A PARCELLÁK1A séma BCNF normalizálásakor az FD2 funkcionális függés eltűnik a felbontásból. (b) Egy sematikus reláció funkcionális függésekkel; 3NF-ben van, de nincs BCNF-ben.



A gyakorlatban a legtöbb relációséma, amely 3NF-ben van, egyúttal BCNF-ben is van. Csak ha $X \rightarrow A$ úgy áll fenn egy *R* relációsémában, hogy *X* nem szuperkulcs *és A* elsődleges attribútum, akkor lesz *R* 3NF-ben úgy, hogy nem lesz BCNF-ben. A 8.12. (b) ábrán bemutatott *R* relációséma illusztrálja egy ilyen relációnak az általános esetét. Ideális esetben a relációsadatbázis-tervnek arra kellene törekednie, hogy elérje a BCNF-et vagy a 3NF-et minden relációséma esetén. A csak 1NF vagy 2NF normalizálási állapot elérését nem tekintjük kielégítőnek, mivel ezek csak köztes lépések a 3NF és BCNF felé.

8.13. ábra - A TANÍT reláció, amely 3NF-ben van, de nincs BCNF-ben.

TANÍT

Hallgató	Kurzus	Gyakvezér
Ablonczy	Adatbázisrendszerek	Kósa
Kovács	Adatbázisrendszerek	Adamkó
Kovács	Operációs rendszerek	Fazekas
Kovács	Programozás	Pánovics
Nemes	Adatbázisrendszerek	Kósa
Nemes	Operációs rendszerek	Balla
Hargitai	Adatbázisrendszerek	Vágner
Molnár	Adatbázisrendszerek	Adamkó
Ablonczy	Operációs rendszerek	Fazekas

Másik példaként tekintsük a 8.13. ábrát, amely a TANÍT relációt mutatja az alábbi függésekkel:

FD1: { Hallgató, Kurzus } → Gyakvezér

FD2:[35] Gyakvezér → Kurzus

Vegyük észre, hogy a { Hallgató, Kurzus } kulcsjelölt a relációban, és hogy a bemutatott függések a 8.12. (b) ábrán látható mintát követik, ahol *A* a Hallgató, *B* a Kurzus és *C* a Gyakvezér. A reláció ezért 3NF-ben van, de nincs BCNF-ben. Ennek a relációsémának a felbontása két sémára nem egyértelmű, mivel az alábbi három lehetséges pár bármelyikére bontható:

1. { Hallgató, Gyakvezér } → { Hallgató, Kurzus }
2. { Kurzus, Gyakvezér } → { Kurzus, Hallgató }
3. { Gyakvezér, Kurzus } → { Gyakvezér, Hallgató }

Mindhárom felbontás hatására *megszűnik* az FD1 funkcionális függés. A felírtak közül a *leginkább megfelelő felbontás* a harmadik lesz, mert az nem állít elő álrekordokat a két reláció összekapcsolása során.

A . alfejezetben az NJB tulajdonságnál leírunk egy tesztet a dekompozíciók nemadditív (veszteségmentes) voltának eldöntésére. Általában egy nem BCNF-ben lévő relációt úgy célszerű szétbontani, hogy megfeleljen ennek a tulajdonságnak, míg esetleg lemondunk arról, hogy a szétbontott relációkban az összes funkcionális függést megőrizzük, akárcsak ebben a példában. Ezt csinálja a 3. algoritmus, s azt használhatjuk, hogy megkapjuk a fenti TANÍT séma harmadik dekompozícióját, amely az alábbi két BCNF relációt eredményezi:

(Gyakvezér, Kurzus) és (Gyakvezér, Hallgató)

Vegyük észre, hogy ha a TANÍT relációban a { Gyakvezér, Hallgató } párt jelöljük ki elsődleges kulcsnak, akkor a Gyakvezér → Kurzus funkcionális függés részleges (nem teljes funkcionális) függést okoz, hiszen a Kurzus az összetett kulcs egy részétől függ. Ezt a funkcionális függést eltávolíthatjuk a 2NF-re alakítás közben, s így is pontosan ugyanazt a két relációt kapjuk eredményül. Ez az eset egy példa arra, amikor végül ugyanahhoz a BCNF tervhez jutunk különböző normalizációs utakon keresztül.

Összefoglalás

Ebben a fejezetben a relációs adatbázis-tervezés néhány buktatójával foglalkoztunk, intuitív érveket felhasználva. Informálisan azonosítottunk néhány mérőszámot egy relációséma *jó* vagy *rossz* voltának a jelzésére, valamint informális irányelveket fektettünk le egy jó terv elkészítéséhez. Ezek az irányelvek azon alapulnak, hogy gondosan elkészítünk egy koncepcionális tervet az ER és EER modellekben, majd helyesen végrehajtjuk a 7. fejezetben leírt leképezési eljárást, hogy relációkra képezzük le az egyedeket és kapcsolatokat. Ezen irányelvek megfelelő betartásával és a

redundancia minimalizálásával elkerülhetjük a beszúrási/törlési/módosítási anomáliákat és a hamis adatok előállítását. A NULL értékek korlátozását javasoltuk, mivel azok problémákat okozhatnak a szelekciós, az összekapcsolási és a csoportosítási műveletek közben. Ezután néhány formális fogalmat vezettünk be, amelyek segítségével felülről lefelé módon, a relációk egyenkénti elemzésével végezhetjük a relációs tervezést. Az analízis és dekompozíció alapú tervezésnek ezt a folyamatát a normalizálási folyamat leírásával definiáltuk.

Definiáltuk a funkcionális függés fogalmát, és ismertettük néhány tulajdonságát. A funkcionális függések egy relációséma attribútumai között fennálló szemantikai megszorításokat adnak meg. Megmutattuk, hogyan lehet funkcionális függések egy adott halmazából további függéseket levezetni levezetési szabályok segítségével. Definiáltuk a lezárt és a lefedés fogalmakat a funkcionális függésekhez kötődően. Ezután definiáltuk egy függéshalmaz minimális lefedését, és megadtunk egy algoritmust a minimális lefedés kiszámítására. Azt is megmutattuk, hogyan ellenőrizhetjük, hogy két funkcionálisfüggés-halmaz ekvivalens-e egymással.

Ezt követően leírtuk a normalizálási folyamatot, amelyet arra használhatunk, hogy jó terveket kapjunk úgy, hogy bizonyos nemkívánatos, *problémás* funkcionális függések létezését teszteljük az egyes relációsémákban. Olyan megoldást javasoltunk, amely minden egyes relációban egy előre rögzített elsődleges kulcson alapuló, egymást követő normalizálási lépések végrehajtásából állt, majd enyhítettük ezt a követelményt, és általánosabb definíciót adtunk a második normálformára (2NF) és a harmadik normálformára (3NF), amelyek a relációk összes kulcsjelöltjét figyelembe veszik. Példákat hoztunk annak illusztrálására, hogy a 3NF általános definíciójának felhasználásával hogyan elemezhetünk és bonthatunk fel egy adott relációt, hogy végül 3NF-ben lévő relációkat kapjunk eredményül.

Végül bevezettük a Boyce–Codd-normálformát (BCNF), s megtárgyaltuk, hogy miért tekinthető a 3NF egy erősebb formájának. Azt is megmutattuk, hogy hogyan kell felbontani egy nem BCNF-ben lévő relációt, figyelembe véve a nemadditív dekompozíciós követelményt.

A 9. fejezet bemutatja a szintézist, valamint a funkcionális függéseken alapuló relációs adatbázis-tervezéshez használt dekompozíciós algoritmusokat. A dekompozícióhoz kötődően ismertetjük a *nemadditív (veszteségmentes) join* és a *függésmegőrzés* fogalmakat, amelyeket bizonyos algoritmusok kikényszerítenek. A 9. fejezet további témakörei közé tartoznak a többértékű függések, a join függések, illetve a negyedik és ötödik normálformák, amelyek ezeken a függéseken alapulnak.

Áttekintő kérdések

1. Mutassa meg, hogy az attribútumok szemantikája a relációséma jóságának informális mértéke!
2. Ismertesse a bővítési, törlési és módosítási anomáliákat! Miért tekintjük őket károsnak? Adjon rájuk példákat!
3. Miért célszerű elkerülni a NULL-okat egy relációban, amennyire csak lehetséges? Ismertesse az álrekordok problémáját, és hogy hogyan előzhetjük meg ezt a problémát!
4. Sorolja fel a relációséma-tervezés bemutatott informális irányelveit! Mutassa meg, miért lehet káros, ha nem követjük ezeket az irányelveket!
5. Mi a funkcionális függés? Honnan származhatnak azok az információk, amelyek alapján egy relációséma attribútumai között fennálló funkcionális függéseket definiáljuk?
6. Miért nem következtethetünk automatikusan egy funkcionális függésre egy konkrét relációállapotból?
7. Mi a szerepük az Armstrong-féle levezetési szabályoknak (IR1-től IR3-ig) a relációs tervezés elméletének fejlődésében?

8. Mit értünk az Armstrong-féle levezetési szabályok teljességén és helyességén?
9. Mit értünk funkcionális függések egy halmazának lezártja alatt? Mutasson rá példát!
10. Mikor nevezzük funkcionális függések két halmazát ekvivalensnek? Hogyan dönthetjük el, hogy ekvivalensek-e?
11. Mit értünk funkcionális függések minimális halmaza alatt? Igaz-e, hogy a funkcionális függések bármely halmazához létezik vele ekvivalens minimális halmaz? Az mindig egyértelmű?
12. Mire utal a *nem normalizált reláció* kifejezés? Hogyan alakultak ki az egyes normálformák történetileg az első normálformától a Boyce–Codd-féle normálformáig?
13. Definiálja az első, a második és a harmadik normálformákat, ha csak az elsődleges kulcsokat vesszük figyelembe! Miben különbözik a 2NF és a 3NF általános definíciója – amelyek a relációk összes kulcsát figyelembe veszik – azoktól, amelyek csak az elsődleges kulcsokat veszik figyelembe?
14. Milyen nem kívánt függéseket küszöbölünk ki, ha egy reláció 2NF-ben van?
15. Milyen nem kívánt függéseket küszöbölünk ki, ha egy reláció 3NF-ben van?
16. Definiálja a Boyce–Codd-normálformát! Miben különbözik a 3NF-től? Miért tekintjük a 3NF erősebb formájának?

Feladatok

1. Bizonyítsa be vagy cáfolja meg a funkcionális függésekre vonatkozó következő levezetési szabályokat. Egy bizonyítást vagy direkt módon történő érveléssel, vagy a reflexivitás, az augmentivitás és a tranzitivitás szabályának a felhasználásával adjon meg. Cáfolatot olyan relációpéldány megadásával tehet, amelyben teljesülnek a levezetési szabály bal oldalán álló feltételek és funkcionális függések, de nem teljesülnek a jobb oldalon állók.
    a. $\{ W \to Y, X \to Z \} \models \{ WX \to Y \}$
    b. $\{ X \to Y \}$ és $Y \supseteq Z \models \{ X \to Z \}$
    c. $\{ X \to Y, X \to W, WY \to Z \} \models \{ X \to Z \}$
    d. $\{ XY \to Z, Y \to W \} \models \{ XW \to Z \}$
    e. $\{ X \to Z, Y \to Z \} \models \{ X \to Y \}$
    f. $\{ X \to Y, XY \to Z \} \models \{ X \to Z \}$
    g. $\{ X \to Y, Z \to W \} \models \{ XZ \to YW \}$
    h. $\{ XY \to Z, Z \to X \} \models \{ Z \to Y \}$
    i. $\{ X \to Y, Y \to Z \} \models \{ X \to YZ \}$
    j. $\{ XY \to Z, Z \to W \} \models \{ X \to W \}$
2. Tekintse a funkcionális függések következő két halmazát: $F = \{ A \to C, AC \to D, E \to AD, E \to H \}$ és $G = \{ A \to CD, E \to AH \}$. Ellenőrizze, hogy ekvivalensek-e egymással!
3. Mutassa meg, hogy $AB \to D$ benne van a $\{ AB \to C, CE \to D, A \to E \}$ függéshalmaz lezártjában!
4. Tekintse a következő relációsémát:
    a. *R*(Orvos#, Beteg#, Dátum, Diagnózis, KezelésKód, Költség)

    Egy ilyen sémájú relációban egy rekord egy betegnek egy orvosnál tett látogatását írja le a

kezelés kódjával és napi költségével együtt. Tegyük fel, hogy az orvos betegenként (egyértelműen) meghatározza a diagnózist. Tegyük fel továbbá, hogy minden kezelési kódhoz egy állandó költség tartozik (tekintet nélkül a betegre). Igaz-e, hogy a relációséma 2NF-ben van? Indokolja válaszát, és bontsa fel a sémát, ha szükséges! Ezután érveljen, hogy szükséges-e további normalizálást végezni 3NF-be vagy sem, és ha igen, hajtsa végre!

[26] Egy kivétel, amelynél mégis ezt a megközelítést alkalmazzák a gyakorlatban, azon a modellen alapul, amelyet bináris relációs modellnek hívnak. Példa rá a NIAM módszer (Verheijen és van Bekkum, 1982).

[27] Ezeket az anomáliákat Codd (Codd (1972)) azonosította, hogy legalizálja a relációk normalizálásának szükségességét, amelyről a . alfejezetben fogunk beszélni.

[28] Ez nem annyira komoly probléma, mint a többi, mivel az összes rekordot egyetlen SQL utasítással módosíthatjuk.

[29] A több alaprelációt összekapcsoló nézeteken alapuló lekérdezések hatékonysága attól függ, hogy a DBMS hogyan implementálja a nézetet. Sok RDBMS materializálja a gyakran használt nézeteket, és így nem olyan gyakran kell végrehajtaniuk az összekapcsolásokat. A DBMS felelős azért, hogy a materializált nézeteket aktualizálja (azonnal vagy periodikusan), valahányszor az alaprelációk módosulnak.

[30] Ez azért van, mert a belső és külső összekapcsolások különböző eredményt adnak, ha NULL-ok is érintettek az összekapcsolásban. A felhasználóknak ezért tisztában kell lenniük a különböző összekapcsolás-típusok jelentésével. Bár a hozzáértő felhasználók számára ez ésszerű lehet, mások számára azonban nem feltétlenül az.

[31] Ez egy standard alak, amivel egyszerűsíthetjük azokat a feltételeket és algoritmusokat, amelyek a redundanciák létezését vizsgálják *F*-ben. Az IR4 levezetési szabály segítségével az olyan függések, amelyeknek a jobb oldalán több attribútum szerepel, átkonvertálhatók olyan függések halmazává, amelyeknek a jobb oldalán csak egy attribútum szerepel.

[32] Ezt a feltétel már nem szerepel a *beágyazott relációs modellben* és az *objektum-relációs rendszerekben* (ORDBMS-ekben), amelyek megengedik a *nem normalizált* relációkat.

[33] Ebben az esetben az Ohelyszínek tartományát egy egyszerű helyszínekből álló halmaz **hatványhalmazának** tekinthetjük; azaz a tartomány az egyszerű helyszínek halmazának összes lehetséges részhalmazából áll.

[34] Ez a tranzitív függés általános definíciója. Mivel ebben a fejezetben csak az elsődleges kulcsokkal foglalkozunk, megengedünk olyan tranzitív függéseket, amelyekben *X* az elsődleges kulcs, míg *Z* lehet kulcsjelölt vagy annak egy részhalmaza.

[35] Ez a függés azt jelenti, hogy az alkalmazásunkban egy megszorítás az is, hogy *minden gyakorlatvezető csak egy kurzust vezet*.

9. fejezet - Relációsadatbázis-tervezési algoritmusok és további függések

Tartalom

A relációs dekompozíciók tulajdonságai

Relációk dekompozíciója és a normálformák elégtelensége

A dekompozíciók függésmegőrző tulajdonsága

A dekompozíciók nemadditív (veszteségmentes) join tulajdonsága

A bináris dekompozíciók nemadditív join tulajdonságának tesztelése

Egymás után alkalmazott nemadditív join dekompozíciók

Algoritmusok a relációs adatbázisséma tervezéséhez

Nemadditív join dekompozíció BCNF sémákra

3. algoritmus − Relációsémák nemadditív join tulajdonságú dekompozíciója BCNF-re.

Bemenet: Egy R univerzális relációséma és az R attribútumain értelmezett funkcionális függéseknek egy F halmaza.

1. Legyen $D := \{ R \}$;
2. Amíg van olyan Q relációséma D-ben, ami nincs BCNF-ben

- {
    - válasszunk egy Q relációsémát D-ben, ami nincs BCNF-ben;
    - keressünk Q-ban egy olyan $X \rightarrow Y$ funkcionális függést, amely megsérti a BCNF-et;
    - helyettesítsük D-ben Q-t a $(Q-Y)$ és az $(X \cup Y)$ relációsémákkal;
- };

Többértékű függések és a negyedik normálforma

9.1. ábra - Negyedik és ötödik normálformák. (a) A DOLG reláció két többértékű függéssel: Dnév → Rnév és Dnév → Hnév. (b) A DOLG reláció felbontása két 4NF relációba: DOLG_PROJEKTEK és DOLG_HOZZÁTARTOZÓK. (c) A többértékű függés nélküli SZÁLLÍTÁS reláció 4NF-ben van, de nincs 5NF-ben, ha rendelkezik a JD(R_1, R_2, R_3) kapcsolásfüggéssel. (d) A SZÁLLÍTÁS reláció felbontása az R_1, R_2, R_3 5NF relációkra.

(a) **DOLG**

Dnév	Pnév	Hnév
Kovács	X	János
Kovács	Y	Anna
Kovács	X	Anna
Kovács	Y	János

(b) **DOLG_PROJEKTEK**

Dnév	Pnév
Kovács	X
Kovács	Y

DOLG_HOZZÁTARTOZÓK

Dnév	Hnév
Kovács	János
Kovács	Anna

(c) **SZÁLLÍTÁS**

Bnév	Alkatrész_név	Proj_név
Kovács	csavar	X projekt
Kovács	anyacsavar	Y projekt
Vágvölgyi	csavar	Y projekt
Lakatos	anyacsavar	Z projekt
Vágvölgyi	szög	X projekt
Vágvölgyi	csavar	X projekt
Kovács	csavar	Y projekt

(d) R_1

Bnév	Alkatrész_név
Kovács	csavar
Kovács	anyacsavar
Vágvölgyi	csavar
Lakatos	anyacsavar
Vágvölgyi	szög

R_2

Bnév	Proj_név
Kovács	X projekt
Kovács	Y projekt
Vágvölgyi	Y projekt
Lakatos	Z projekt
Vágvölgyi	X projekt

R_3

Alkatrész_név	Proj_név
csavar	X projekt
anyacsavar	Y projekt
csavar	Y projekt
anyacsavar	Z projekt
szög	X projekt

A többértékű függés formális definíciója

Definíció. Egy R relációsémán megadott $X \rightarrow Y$ többértékű függés, ahol X és Y az R attribútumhalmazai, a következő megszorítást jelenti bármely R fölötti r reláció esetén: Ha van két olyan t_1 és t_2 rekord r-ben, amelyre $t_1[X] = t_2[Y]$, akkor léteznie kell két t_3 és t_4 rekordnak is r-ben a következő tulajdonságokkal, ahol Z-t az $(R - (X \cup Y))$ jelölésére használjuk:

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
- $t_3[Y] = t_1[Y]$ és $t_4[Y] = t_2[Y]$.
- $t_3[Z] = t_2[Z]$ és $t_4[Z] = t_1[Z]$.

Valahányszor $X \rightarrow Y$ fennáll, azt mondjuk, hogy X **többértékűen meghatározza** Y-t. A definícióbeli szimmetria miatt valahányszor $X \rightarrow Y$ fennáll R-ben, mindannyiszor $X \rightarrow Z$ is teljesül. Így az $X \rightarrow Y$ többértékű függésből következik $X \rightarrow Z$, és ezért ezt néha úgy szoktuk írni, hogy $X \rightarrow Y|Z$.

Funkcionális és többértékű függések levezetési szabályai

A negyedik normálforma

Nemadditív join dekompozíció 4NF relációsémákra történő felbontáshoz

Valahányszor felbontunk egy R relációsémát az R-en fennálló $X \rightarrow Y$ többértékű függés alapján $R_1 = (X \cup Y)$ és $R_2 = (R - Y)$ relációsémákra, a dekompozíció nemadditív join tulajdonságú. Meg lehet mutatni, hogy ez egy szükséges és elégséges feltétele egy séma két olyan sémára történő felbontásának, amelyek nemadditív join tulajdonságúak, ahogyan azt az NJB' tulajdonság is kimondja, amely a korábban megadott NJB tulajdonságnak egy további általánosítása.

NJB' tulajdonság. Az R_1 és R_2 relációsémák akkor és csak akkor alkotják R egy nemadditív join tulajdonságú dekompozícióját, figyelembe véve a funkcionális és többértékű függések egy F halmazát, ha

$(R_1 \cap R_2) \rightarrow (R_1 - R_2)$

vagy — szimmetrikusan — akkor és csak akkor, ha

$(R_1 \cap R_2) \rightarrow (R_2 - R_1)$.

A 3. algoritmus egy kis módosításával kapjuk a 5. algoritmust, amely végrehajt egy nemadditív join dekompozíciót, olyan relációsémákat előállítva, amelyek 4NF-ben vannak (BCNF helyett). Ahogyan a 3. algoritmus, úgy a 5. algoritmus *sem* szükségképpen ad olyan dekompozíciót, amely megőrzi a funkcionális függéseket.

5. algoritmus − Relációsémák nemadditív join tulajdonságú dekompozíciója 4NF relációsémákra.

Bemenet: Egy R univerzális relációséma és az R attribútumain értelmezett funkcionális és többértékű függéseknek egy F halmaza.

1. Legyen $D := \{ R \}$;
2. Amíg van olyan Q relációséma D-ben, ami nincs 4NF-ben

- {
    - válasszunk egy Q relációsémát D-ben, ami nincs 4NF-ben;
    - keressünk Q-ban egy olyan nemtriviális $X \rightarrow Y$ többértékű függést, amely megsérti a 4NF-et;
    - helyettesítsük D-ben Q-t a $(Q-Y)$ és az $(X \cup Y)$ relációsémákkal;
- };

Kapcsolásfüggések és az ötödik normálforma

Láttuk, hogy az NJB és az NJB' adják meg azt a feltételt, hogy egy R relációséma mikor bontható szét R_1 és R_2 sémákra úgy, hogy a felbontás rendelkezzen a nemadditív join tulajdonsággal.

Bizonyos esetekben azonban elképzelhető, hogy nincs olyan nemadditív join dekompozíció, amely R-et *két* relációsémára bontja, de létezhet olyan nemadditív join dekompozíció, amely *kettőnél több* relációsémát eredményez. Ráadásul az is elképzelhető, hogy nincs olyan funkcionális függés R-en, amely bármely normálformát sértene a BCNF-ig, és nincs olyan nemtriviális többértékű függés sem R-en, amely a 4NF-et sértené. Ebben az esetben egy újabb függésfajtát veszünk igénybe, amelyet *kapcsolásfüggésnek* nevezünk, és ha az fennáll, akkor egy *többágú dekompozíciót* hajtunk végre az

ötödik normálforma (5NF) eléréséhez. Fontos megjegyezni, hogy ez a függés egy nagyon speciális szemantikai megszorítás, amelyet a gyakorlatban nagyon nehéz felfedezni; ezért a gyakorlatban az 5NF-be történő normalizálást nagyon ritkán végzik el.

Definíció. Egy R relációsémán megadott **kapcsolásfüggés (join dependency, JD)** meghatároz egy megszorítást az R bármely r relációjára. A megszorítás azt írja elő, hogy R minden legális r relációjának kell, hogy legyen egy veszteségmentes join dekompozíciója az R_1, R_2, … R_n sémákba; azaz minden ilyen r-re

$*(\pi_{R1}(r), \pi_{R2}(r), \ldots, \pi_{Rn}(r)) = r$.

Az így előírt megszorítást JD(R_1, R_2, … R_n)-nel jelöljük.

A többértékű függés olyan speciális esete a kapcsolásfüggésnek, amikor $n = 2$. Azaz egy JD(R_1, R_2) implikál egy $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ többértékű függést (illetve szimmetrikusan egy $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ többértékű függést is). Egy R séma feletti JD(R_1, R_2, … R_n) kapcsolásfüggés **triviális** kapcsolásfüggés, ha valamely JD(R_1, R_2, … R_n)-beli R_i relációséma egyenlő R-rel. Az ilyen függést azért nevezzük triviálisnak, mert az R bármely r relációállapotán rendelkezik a nemadditív join tulajdonsággal, és így nem határoz meg semmilyen megszorítást R-en. Ezek után definiálhatjuk az ötödik normálformát.

Definíció. Egy R relációséma **ötödik normálformában (5NF-ben)** van, figyelembe véve funkcionális, többértékű és kapcsolásfüggések egy F halmazát, ha minden F^+-beli (azaz F-ből következő) nemtriviális JD(R_1, R_2, … R_n) esetén minden R_i szuperkulcsa R-nek.

A kapcsolásfüggésre példaként tekintsük még egyszer a 9.1. (c) ábrán látható SZÁLLÍTÁS csupakulcs relációt. Tételezzük fel, hogy fennáll a következő megszorítás: Ha egy *b* beszállító szállít egy *a* alkatrészt, *és* egy *p* projekt felhasználja az *a* alkatrészt, *és* a *b* beszállító *legalább egy* alkatrészt szállít a *p* projekthez, *akkor* a *b* beszállító az *a* alkatrészt is szállítja a *p* projekthez. Ezt a megszorítást más módokon is megfogalmazhatjuk, és egy JD(R_1, R_2, R_3) kapcsolásfüggést határoz meg a SZÁLLÍTÁS R_1(Bnév, Alkatrész_név), R_2(Bnév, Proj_név) és R_3(Alkatrész_név, Proj_név) projekcióin. Ha ez a megszorítás fennáll, akkor a 9.1. (c) ábrán a szaggatott vonal alatti rekordoknak léteznie kell a SZÁLLÍTÁS relációséma minden olyan legális állapotában, amelyben a szaggatott vonal feletti rekordok léteznek. A 9.1. (d) ábra azt mutatja, hogy a *kapcsolásfüggéssel rendelkező* SZÁLLÍTÁS relációséma hogyan bontható fel az R_1, R_2 és R_3 relációsémákra, amelyek 5NF-ben vannak. Tekintve ennek a három relációsémának egy-egy relációpéldányát, megfigyelhetjük, hogy az ezen relációk közül bármely kettőre alkalmazott természetes összekapcsolás *álrekordokat eredményez*, míg a *mindhárom* relációra alkalmazott természetes összekapcsolás nem. Az olvasó ellenőrizheti ezt a 9.1. (c) ábrán látható példa reláción, illetve annak a 9.1. (d) ábrán látható projekcióin. Ezért azért van, mert csak a kapcsolásfüggés létezik, de nincs többértékű függés. Azt is vegyük észre, hogy a JD(R_1, R_2, R_3) *minden* legális relációállapoton érvényes, nem csak a 9.1. (c) ábrán láthatón.

Az attribútumok százaival rendelkező valós világbeli adatbázisokban gyakorlatilag lehetetlen felfedezni a kapcsolásfüggéseket. Ez csak akkor tehető meg, ha a tervező kellő mélységben átlátja az adatok közötti összefüggéseket. Emiatt manapság az adatbázis-tervezési gyakorlat minimális figyelmet szentel nekik.

Tartalmazásfüggés

További függések és normálformák

Összefoglalás

Áttekintő kérdések

Feladatok

Irodalomjegyzék

[Abrial (1974)] Abrial, J.. 1974. *Data Semantics*. In Klimbie and Koffeman [1974].

[Codd (1972)] Codd, E.. 1972. *Further Normalization of the Data Base Relational Model*. Rustin.

[Elmasri et al. (1985)] Elmasri, R., Weeldreyer, J., és Hevner, A.. May 1985. *The Category Concept: An Extension to the Entity-Relationship Model*. DKE. 1. 1.

Tárgymutató

Jelzések

A

B

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V