ADATBÁZIS-KEZELÉS

Ajánlott irodalom:

Békési – Geda – Holovács – Perge : Adatbázis kezelés

Főiskolai jegyzet (Eger, Líceum kiadó)

Bódy Bence: Az SQL példákon keresztül

Jedlik Oktatási Stúdió

Joe Celko: SQL fejtörők

Kiskapu kiadó

Az adatbázis-kezelés alapjai

Adat és információ

Adatbázisok

Ahhoz hogy az adatokat egy számítógépes rendszerben tárolni tudjuk, valamilyen struktúrába kell őket szervezni. Ezt általában a köztük lévő logikai összefüggések alapján szoktuk megtenni.

Az integrált, logikailag összetartozó, hosszú ideig tárolt adatok, információk összességét *adatbázisnak* nevezzük.

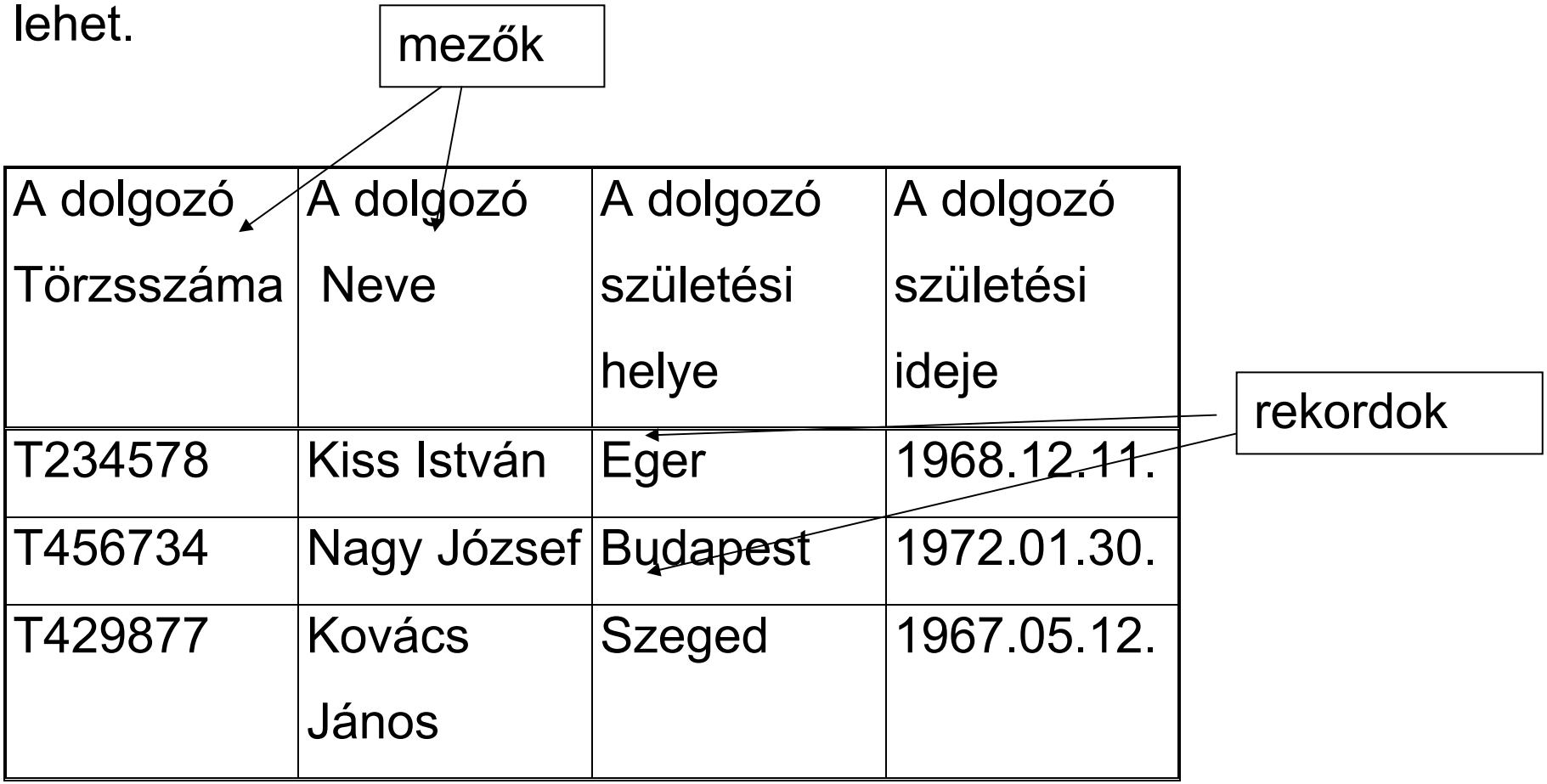
Adatbázis-rendszerek

magába foglalják az adatbázisokat, a hozzájuk kapcsolódó számítógépes erőforrásokat, s tágabb értelemben az adatbázisok tervezését, kezelését végző személyeket is. Ez utóbbiakat nevezzük: Adatbázis adminisztrátoroknak.

Minden adatbázisnak van egy belső struktúrája ***sémája***. Ez tartalmazza az összes adatelem és a köztük lévő kapcsolatok definícióját, leírását. Az adatbázisnak az egyes személyek szempontjából tekintett sémáját ***alsémának*** nevezzük. **Metaadatok**

Példa

Egy vállalat dolgozóinak adatait nyilvántartó program adatbázisának része lehet.



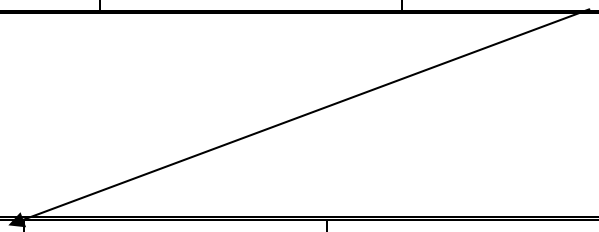
A dolgozó Törzsszáma	A dolgozó Neve	A dolgozó születési helye	A dolgozó születési ideje
T234578	Kiss István	Eger	1968.12.11.
T456734	Nagy József	Budapest	1972.01.30.
T429877	Kovács János	Szeged	1967.05.12.

Adatredundancia

Adatredundancián azt értjük, ha egy adatot egynél több helyen tárolunk egy számítógépes rendszerben. Azt nehéz elkerülni, hogy redundancia egyáltalán ne forduljon elő, azonban a többszörös előfordulások minimálisra csökkentése minden esetben fontos cél. Például ha egy adat több helyen szerepel, és azt módosítjuk, akkor az összes előfordulást módosítani kell. A redundancia kiküszöbölésének szokásos módszere, hogy az adatbázis tervezése során az ismétlődő adatokat „kiemeljük” és külön helyen tároljuk, a megfelelő helyen hivatkozva rá.

Példa

A kifizetés dátuma	Kifizetett Bér	Levont adóelőleg	Hivatkozás a dolgozóra



A dolgozó Törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje

Adatbázis-kezelő rendszerek

Az AB kezelésére speciálisan erre a célra kifejlesztett programok léteznek:

Adatbázis-kezelő rendszernek (ABKR).

Az ABKR nem felhasználói programnak tekinthető, mivel fő feladatai nem egyedi felhasználói igényeket elégítsenek ki. Ez utóbbiak megvalósításához külön alkalmazásokat szokás fejleszteni, az ABKR által nyújtott eszközöket felhasználva.

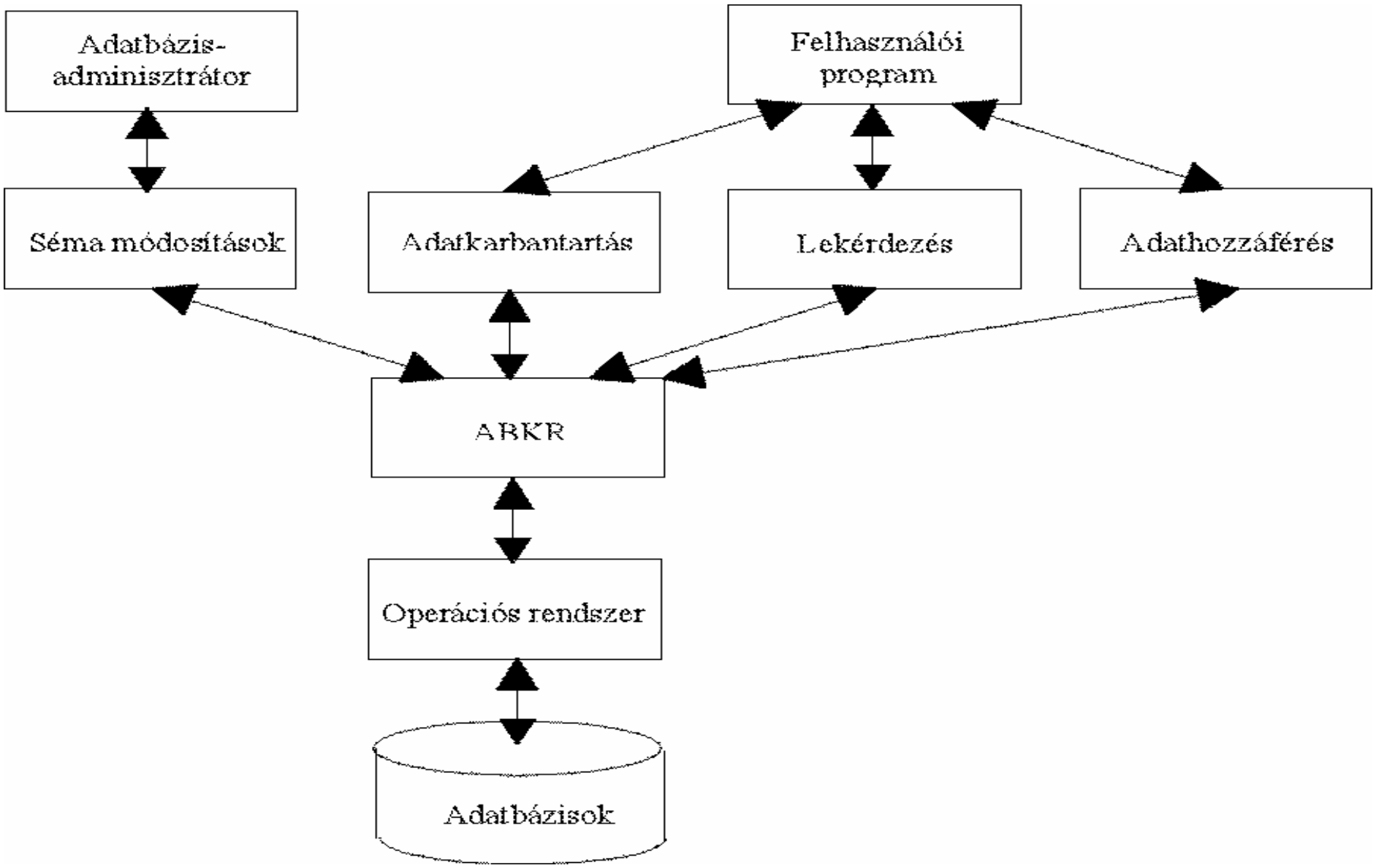
Egy ABKR általában a következő feladatokat látja el:

- Támogatja új adatbázisok létrehozását, azok struktúrájának, tárolási módjának kialakítását.
- Megvalósítja az adatbázisban tárolt adatok kezelését, karbantartását. Ennek keretében új adatok tárolhatók, a tárolt adatok módosíthatók, törölhetők.
- Lehetővé teszi a tárolt adatok feldolgozását, lekérdezését. Ennek keretében új információkat képes előállítani, illetve a meglévő információknak a felhasználók által igényelt formában történő megjelenítését támogatja.
- Garantálja az adatok biztonságát, konzisztenciáját, a hozzáférések szabályozását, vagyis hogy a felhasználói műveleteket csak arra

jogosult személyek végezhessék, és ezek a műveletek ne veszélyeztessék az adatok integritását.

- Lehetővé teszi az adatbázisok megosztását több felhasználó között.

A következő ábra az ABKR helyét mutatja be a számítógépes rendszerben.



Az **ANSI/SPARC modell**

Amerikai Szabványügyi Hivatal (ANSI=American National Standards Institute) Szabvány Tervezési és Követelmények Bizottsága (SPARC=Standards Planning And Requirements Committe) ezt az összeköttetést három szintre osztotta fel.

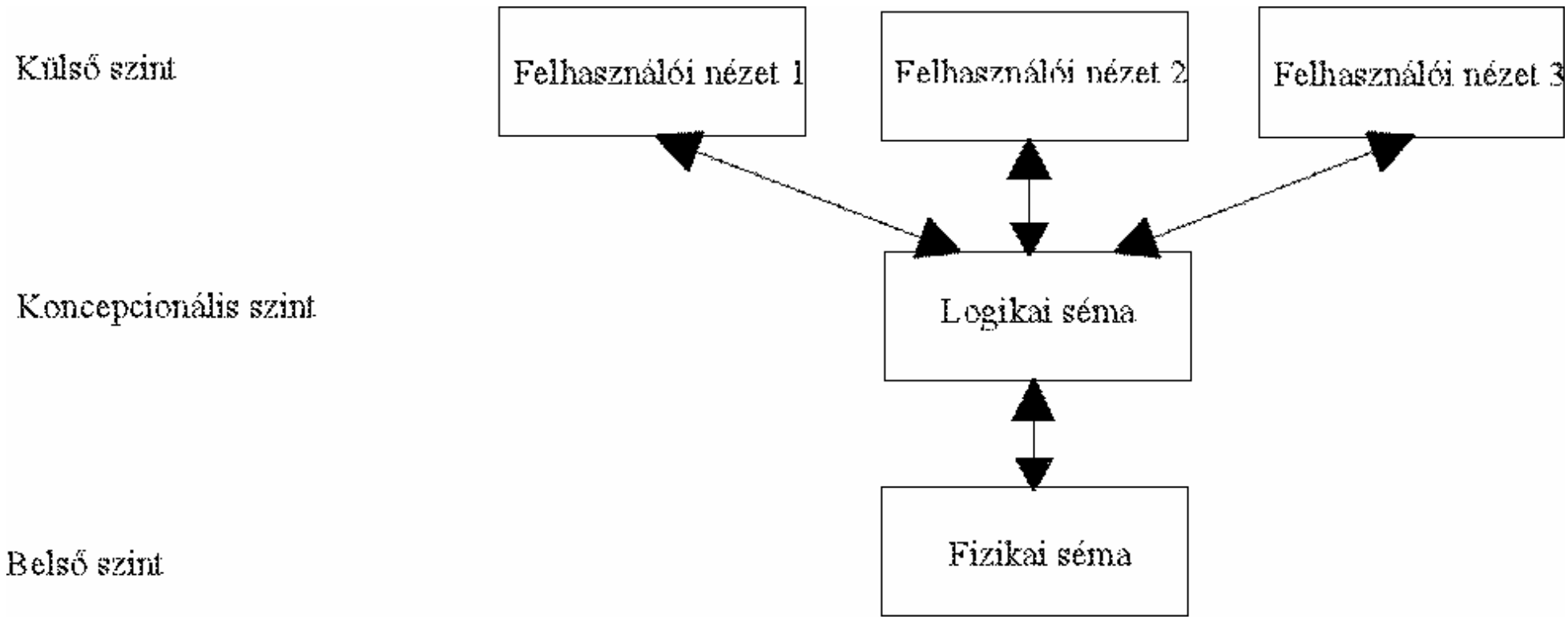
Ezek a ***külső*, a *koncepcionális* és a *belső* szintek.**

A **külső szint** a felhasználó szemszögéből vizsgálja az adatokat. Az adatbázis tartalma ezen a szinten **jelentések, űrlapok,** v. más **dokumentumok** formájában jelenik meg. Az adatok azon nézetét, ahogyan a különböző felhasználók látják azokat ***felhasználói nézetnek (user view)*** nevezik.

A középső szinten a **koncepcionális szint.**

Ez az a szint magában foglalja az összes felhasználói nézetet, így tartalmazza mindazokat az adatokat, amelyre valamely felhasználónak szüksége lehet. Ezen a szinten az adatbázist az úgynevezett ***logikai sémával*** szokás megadni.

A harmadik szint a **belső szint**. Az adatoknak a számítógépes rendszerben való aktuális reprezentációját jelenti. Szokás ezt ***fizikai szintnek*** is nevezni, az adatok reprezentációját pedig ***fizikai sémának*.**


Külső szint
Felhasználói nézet 1
Felhasználói nézet 2
Felhasználói nézet 3
Koncepcionális szint
Logikai séma
Belső szint
Fizikai séma

***Fizikai adatfüggetlenség*.** Azt jelenti, hogy a fizikai sémának a változása nincs hatással a felette lévő szintekre. Így a belső szinten történő változások végrehajthatók anélkül, hogy módosítani kellene a logikai sémát.

***Logikai adatfüggetlenség*.** A felhasználó hozzáadhat új nézetet a rendszerhez anélkül, hogy ez változtatást igényelne a logikai sémában.

Adatbázisok felépítése, adatmanipuláció

Azt a nyelvet, amelynek segítségével az adatbázis adminisztrátorok az új adatbázisok sémáját definiálhatják, ***adatdefiníciós nyelvnek (DDL=Data Definition Language)*** nevezzük.

Adatmanipulációs nyelvnek (DML=Data Manipulation Language).

L*ekérdező nyelvnek (QL=Query Language)*. Ezek a nyelvek általában kevésbé bonyolultak, mint egy általános célú programozási nyelv. Legtöbb esetben nem tartalmaznak vezérlő utasításokat és adatszerkezetek kezelését sem támogatják. Az utasítások csupán az adatbázisban tárolt adatok manipulálását, hozzáférését szolgálják.

A leggyakrabban használt művelet, amit egy ABKR az adatmanipulációk során használ a **keresés.**

Az egyik leggyakrabban alkalmazott, keresést támogató adatstruktúra a ***B-fa*.** Nevében a B betű a **kiegyensúlyozott (balanced)** szó angol megfelelőjére utal. Így egy kereséshez minimális lemezművelet szükséges.

Az adatbázisoknál alkalmazott, a keresést támogató adatstruktúrát ***indexnek*** nevezzük.
Az adatbázis sémájánál meg kell adni, hogy mely mezők alapján kívánunk indexeket készíteni. Az index arra szolgál, hogy a keresési és rendezési

műveleteket gyorsabbá tegye. De a beszúrás, törlés, módosítás műveleteknél inkább lassító hatása van, mivel ezeknél az index adatstruktúrát is aktualizálni kell. Használata olyan adatbázisoknál előnyös, amelyeknél kevés módosítás történik, viszont gyakran kell keresni.

A korszerű ABKR-ek lehetőséget biztosítanak felhasználói felületek tervezésére és kivitelezésére.

Képernyőtervező (screen generator).

Segítségével a felhasználó, vagy a programozó különböző *űrlapokat*, vagy *beviteli képernyőket* tervezhet és állíthat elő, melyekkel az adatbázis adatait lehet bővíteni, módosítani, törölni.

Jelentéstervezőnek (report generator)

Az ABKR-ek rendelkeznek olyan modullal, amelynek segítségével nyomtatott jelentések tervezhetők. A jelentéstervezők speciális parancsokkal rendelkeznek, amelyekkel címeket, fejléceket, sorokat, oszlopokat, összegeket és más, jelentésben gyakran előforduló elemeket alakíthatunk ki.

ABKR, amelyek segítségével gyorsan és kényelmesen fejleszthetünk adatbázis-alkalmazásokat ***4GL (negyedik generációs nyelv)*** rendszereknek hívjuk.

A 4GL rendelkezik képernyőtervezővel, amivel az adatkarbantartás könnyen végezhető.

A 4GL rendelkezik jelentéstervezővel, amivel program készítése nélkül gyorsan tudunk listákat készíteni.

A 4GL támogatja eljárások megtervezését képernyő segítségével – programozás nélkül – és ezekből programkódot is képes generálni.

Manapság a 4GL-ek nagy része grafikus felülettel rendelkező operációs rendszeren fut, így a fenti elveket grafikus megjelenítés segítségével valósítja meg.

Adatintegritás

Az adatok korrektek, konzisztensek és aktuálisak.

Legtöbbször már a séma definiálásakor megadhatók kritériumok, feltételek az egyes adatok tartalmára és formátumára vonatkozóan. Az adat bevitelekor az ABKR ellenőrzi, hogy az aktuális adat teljesíti-e ezeket a követelményeket. Amennyiben nem, az adatot nem fogadja el.

Az adatbázis a tényleges adatok mellett tárolja azok összefüggéseit, kapcsolatait is.

Hivatkozási integritás.

Ez azt jelenti, hogy egy kapcsolatnál a hivatkozott adatnak létezni kell. A korszerű ABKR az adatmanipulációk során ellenőrzi, hogy a hivatkozási integritást nem sérti-e a művelet.

Az adatok helyessége nemcsak az ABKR felelőssége, az alkalmazás fejlesztőjének is igyekeznie kell úgy megterveznie az adatbázist, hogy abban a lehető legkevesebb lehetőség legyen arra, hogy az adatok valamilyen szempontból helytelenek legyenek.

Adatvédelem

- Védelem az illegális hozzáféréssel szemben.
- Az adatokban bekövetkező hibák, sérülések kivédése, megakadályozása.

Az adatbiztonság megőrzésének egyik lehetséges technikája a ***tranzakció vezérlés***. Ez vázlatosan azt jelenti, hogy a kritikus módosítások végrehajtása az adatbázisban nem közvetlen módon történik, csupán a művelet során regisztrálásra kerül. Amikor a teljes művelet sikeresen befejeződik, csak akkor történnek meg a tényleges módosítások.

Másik fontos technikája az adatvédelemnek a ***biztonsági másolatok (backup copy)*** készítése. Ezeket bizonyos időszakonként lehet elkészíteni. A köztes időszakokban történő változtatásokat a tranzakció vezérléshez hasonlóan külön nyilvántarthatja az ABKR. A biztonsági másolatokra alkalmazva a nyilvántartásban szereplő módosításokat mindig helyreállítható az aktuális állapot, amennyiben meghibásodás történik.

Adatbázisok megosztása

Sok esetben fordul elő, hogy több felhasználó szeretne dolgozni ugyanazon az adatbázison. Ez történhet egyidejűleg, vagy különböző időpontokban is. Az egyidejűleg történő hozzáférés esetében az ABKR-

nek biztosítani kell az adatbázis megosztását a felhasználók között, valamint ügyelnie kell arra, hogy a közös használatból adódó speciális helyzet ellenére az adatbázis konzisztens maradjon. Ez adott esetben nem olyan egyszerű feladat.

Példa

Tegyük fel, hogy egy bérszámfejtő rendszerben ketten dolgoznak. Az egyik felhasználó a fizetésemeléseket rögzíti, a másik pedig éppen egy olyan alkalmazást futtat, ami újraszámolja az adókat. Feltételezzük, hogy a második program kiolvassa a régi fizetést egy adott személy esetén az adatbázisból, s miközben az adószámítási eljárás fut, az első program módosítja a fizetést és az adót is. A második eljárás azonban erről mit

sem tud, tehát amikor kész van, visszaírja az adatbázisba az adót. Ez azonban még a régi fizetés adója, ami kevesebb, mint a tényleges adó.

Kizárólagos joggal való megnyitás.

Több-felhasználós környezetben ez elterjedt megoldás. A **kizárólagos jog** vonatkozhat nagyobb egységekre (például egy **táblára**) – ez akadályozhatja a feldolgozást – vagy kisebb egységekre (például **rekordokra**).

Ha több személy dolgozik egy rendszerben, szokás őket csoportokba sorolni. Az egyes csoportok különböző jogosultságokat kaphatnak az adatbázis egyes részeihez való hozzáféréshez.

Adatmodellezés

Adatbázis tervezés

Az adatbázis tervezés az alkalmazás fejlesztésének egyik kulcskérdése. Nagyobb volumenű alkalmazásoknál általában igen sok adat kerül tárolásra, illetve feldolgozásra. Ezek összetett módon kapcsolódhatnak egymáshoz. Az adatbázisokat úgy kell megtervezni, hogy minimális legyen bennük a redundancia, teljesüljenek a különböző adatfüggetlenségek, stb.

A tervezés maga egy **kreatív folyamat**. Táblázatos formában foglaljuk össze egy alkalmazás tervezésének legfontosabb fázisait.

Fázis neve	**Fő tevékenységek**
Információ igény meghatá-rozása	Célok meghatározása, adatok, formátumok, algoritmusok kialakítása
Logikai adatbázis tervezés	Egyedek meghatározása Egyedeket leíró tulajdonságok megadása Egyedek közötti kapcsolatok feltérképezése Adatredundancia minimalizálása
Fizikai adatbázis kialakítás	Az adatbázisok létrehozása számítógépen

A szervező a tervezés során megismeri a pontos felhasználói igényeket. Különböző meglévő dokumentumok, jelentések gyakran hasznos forrásként szolgálhatnak ehhez.

- A rendelkezésre álló adatokat hogyan fogja a rendszer feldolgozni.
- Az eljárások, algoritmusok specifikációját is ekkor kell elvégezni.
- Meg kell határozni milyen módon és formátumban kívánja a felhasználó megjeleníteni a feldolgozott adatokat.
- Ez befolyásolhatja az adatok tárolási formátumát is, amelyet ugyancsak specifikálni kell.
- Lényeges lehet annak meghatározása is, hogy az egyes adatok feldolgozása mekkora aktivitással történik majd.

A **logikai tervezés** fázisában főképpen az adatokra és a közöttük lévő kapcsolatokra koncentrálunk.

Ebben a lépésben kell részletezni a nyilvántartani kívánt adatok pontos listáját, meghatározni azok tárolási típusát és formátumát, figyelembe véve a számítógép lehetőségeit. Logikai összetartozásuk alapján az adatokat csoportosítani kell, majd meg kell határozni az egyes csoportok közötti összefüggéseket, kapcsolatokat.

Viszonylag jobban elkülönül az előző két fázistól a **fizikai tervezés** szakasza. Ez tulajdonképpen az alkalmazás megvalósítását jelenti a logikai tervek alapján.

A **prototípus** a rendszer első verziója, amely esetleg még nem teljes, illetve finomításra szorul. Alkalmas azonban arra, hogy felújítsa a kommunikációt a felhasználó és a programozó illetve a szervező között, javítva ezzel a rendszer minőségét. Nem feltétlenül szükségszerű a teljes rendszerre vonatkozó prototípus megvalósítása.

Adatmodellezés alapelemei.

Az egységes tervezés érdekében kidolgoztak modelleket, amelyeket ***adatmodelleknek*** szokás nevezni.

Egy adatmodell alapvetően nem magukkal az adatokkal foglalkozik, hanem azok struktúrájával, a közöttük lévő összefüggésekkel, kapcsolatokkal. A legelterjedtebb adatmodellezési technikák közös jellemzője, hogy három alapelemből tevődnek össze. Ezek a következők:

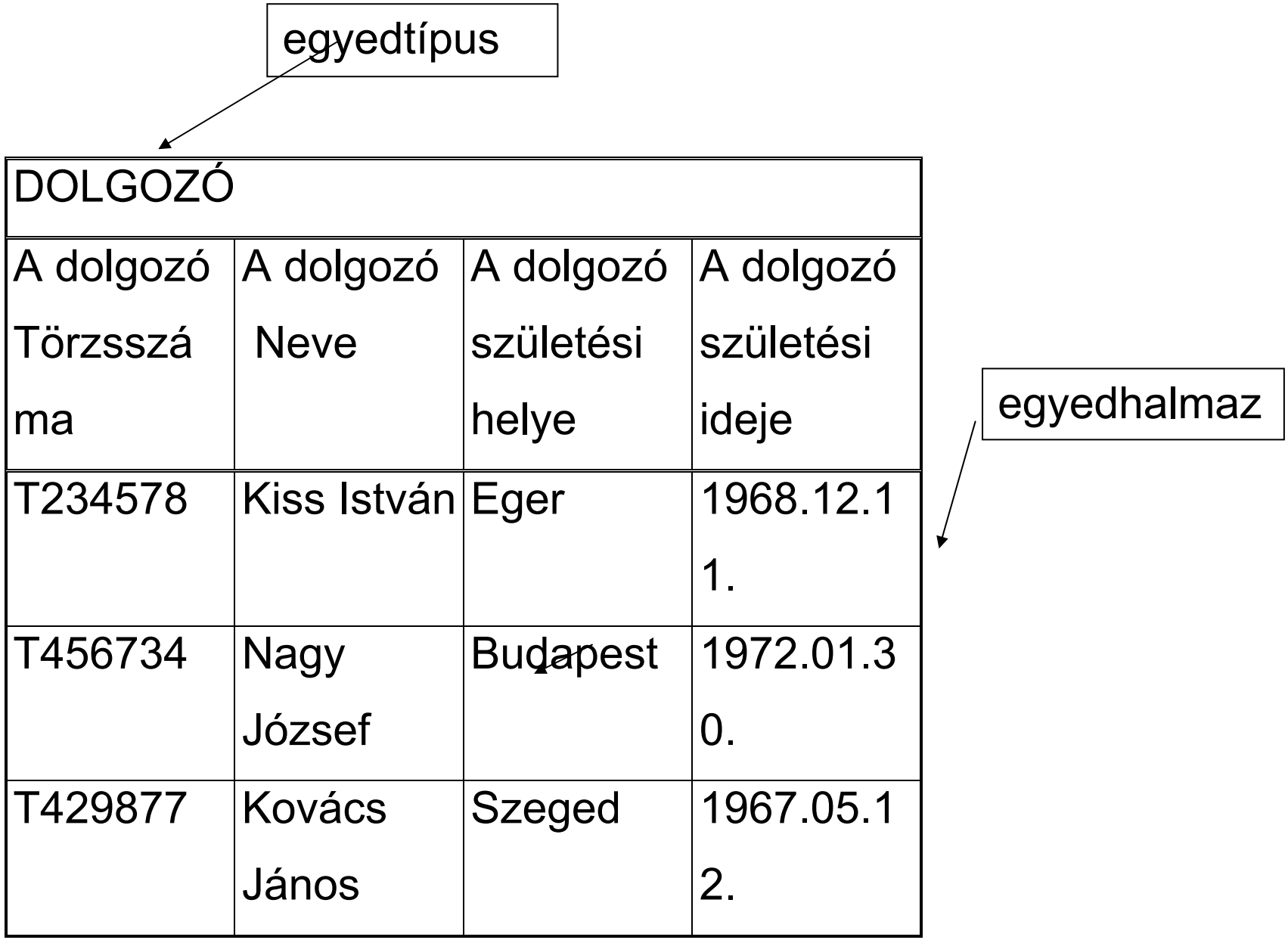
Egyed, vagy egyedtípus

Tulajdonság, vagy tulajdonságtípus

Kapcsolat, vagy kapcsolattípus

Egyed

Egyednek nevezzük azokat a dolgokat, objektumokat, amelyek egymástól jól elkülöníthetők, melyekről adatokat tárolunk és tulajdonságokkal jellemzünk. Egyedek lehetnek például a **dolgozó, kifizetés, anyag, személy,** stb. Ebben a formában az egyed mint absztrakt fogalom szerepel. Mondhatjuk azt is, hogy **az egyed konkrét dolgok absztrakciója.** Az absztrakt egyedekre szokás használni az ***egyedtípus*** kifejezést is. Tekinthetjük azonban az **egyedtípus konkrét előfordulásait** is. Ezeket nevezhetjük ***egyedhalmaznak***, vagy egyszerűen csak ***egyed előfordulásoknak*.** Például a dolgozó egyedtípus egyedhalmaza az összes dolgozót magába foglaló halmaz, hasonlóan a kifizetés egyedhalmaza az összes kifizetés, ami történt.


egyedtípus
DOLGOZÓ
A dolgozó Törzsszáma
A dolgozó Neve
A dolgozó születési helye
A dolgozó születési ideje
T234578
Kiss István
Eger
1968.12.11.
T456734
Nagy József
Budapest
1972.01.30.
T429877
Kovács János
Szeged
1967.05.12.
egyedhalmaz

Az egyed előfordulások a rekordoknak felelnek meg. A gyakorlatban az egyedtípust szokás ***rekordtípusnak*** is nevezni. (rekord- vagy struktúratípus).

Tulajdonság

Az adatmodellben olyan tulajdonságokat szokás tekinteni, amelyek a rendszer szempontjából lényegesek. Például a bérszámfejtő rendszerben a dolgozó neve, fizetése lényeges tulajdonságok.

Az egyed fogalmánál már megismert két szintet itt is alkalmazhatjuk. Maga a tulajdonság egy absztrakt fogalom, amelyet nevezhetünk ***tulajdonságtípusnak***. Az egyes tulajdonságtípusok konkrét értékeit nevezzük ***tulajdonságértékeknek*.** Így például a szín tulajdonság konkrét értékei lehetnek a piros, zöld, kék, stb. Láthatjuk azt is, hogy a különböző tulajdonságok egyes értékeivel a tulajdonságokkal jellemzett egyed egy előfordulását, vagyis az egyedhalmaz egy elemét adhatjuk meg.

DOLGOZÓ			
A dolgozó Törzsszáma	A dolgozó Neve	A dolgozó születési helye	A dolgozó születési ideje
T234578	Kiss István	Eger	1968.12.11.
T456734	Nagy József	Budapest	1972.01.30.
T429877	Kovács János	Szeged	1967.05.12.

tulajdonságtípus

értékek

KULCS. Fontos szerepe van azoknak a tulajdonságoknak, amelynek értékei a többi tulajdonság értékeit egyértelműen meghatározzák.
Ez azt jelenti, hogy ha az ilyen tulajdonságok értékeit megadjuk, akkor az egyértelműen definiál egy előfordulást. Azokat a tulajdonságokat, amelyek egyértelműen meghatározzák az egyedtípus egy elemét, ***kulcsnak*** nevezzük.

Elvileg egy egyednek több kulcsa is lehet, de a legtöbb esetben egyet szokás kiválasztani, amely a leginkább alkalmas az egyértelmű azonosításra. Ezt hívjuk ***elsődleges kulcsnak.***

Kapcsolat

Az adatmodell harmadik fontos elemét a ***kapcsolatok*** jelentik. Kapcsolatnak nevezzük az egyedek közötti összefüggést, viszonyt.

Például a már jól ismert bérszámfejtő rendszerben a dolgozó és a kifizetés egyedek között létezik egy természetes kapcsolat. Ez azt mondja meg, hogy az egyes dolgozókhoz mely kifizetések tartoznak.

A fentiek alapján kapcsolatok az egyedhalmazok elemei között alakíthatók ki.

Osztályozhatjuk a kapcsolatokat aszerint, hogy egy-egy elemhez hány másik elem tartozik.

Egy-egy típusú

Egy-sok típusú

Sok-sok típusú

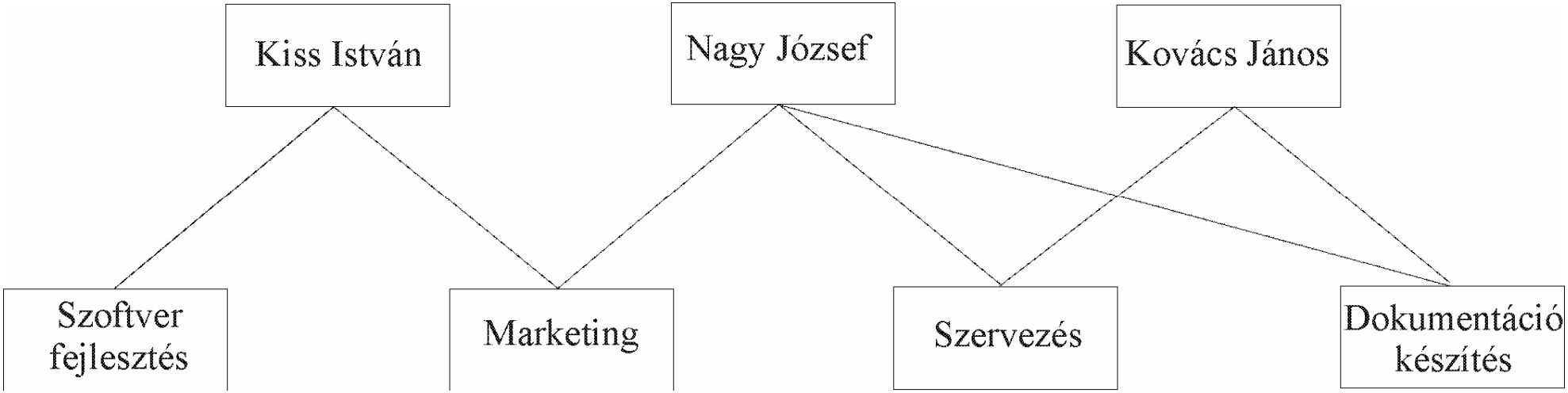
Egy–egy típusú kapcsolat esetén az egyik egyed minden egyes előfordulásának a másik egyed pontosan egy előfordulása tartozik. Egy–egy típusú kapcsolat például a férfi és a nő egyedek között a házastárs kapcsolat.

A következő csoportot az **egy-sok típusú kapcsolatok** alkotják. Ezeknél az egyik egyed minden előfordulásához a másik egyed több előfordulása

tartozhat. Például a bérszámfejtő rendszerben egy-sok kapcsolat van a dolgozók és a kifizetések között.

A kapcsolatok legáltalánosabb formáját a **sok-sok kapcsolatok** jelentik. Sok-sok kapcsolat esetén mindkét egyed előfordulásaihoz a másik egyed több előfordulása tartozhat.

Tegyük fel hogy dolgozói rendszerünkben azt is nyilvántartjuk, hogy melyik dolgozó milyen témákon dolgozik. Egy dolgozó több témában is tevékenykedhet és egy témán több dolgozó dolgozhat. Ebben az esetben tehát sok-sok kapcsolatról van szó. Ezt a következő ábra ezt szemlélteti.



A sok-sok kapcsolatok láthatóan egy-sok kapcsolatokon alapszanak. Ha bármelyik egyed szempontjából nézzük, akkor egy-sok kapcsolatot fedezhetünk fel. **Ezért minden sok-sok kapcsolat felbontható két egy-sok kapcsolatra.**

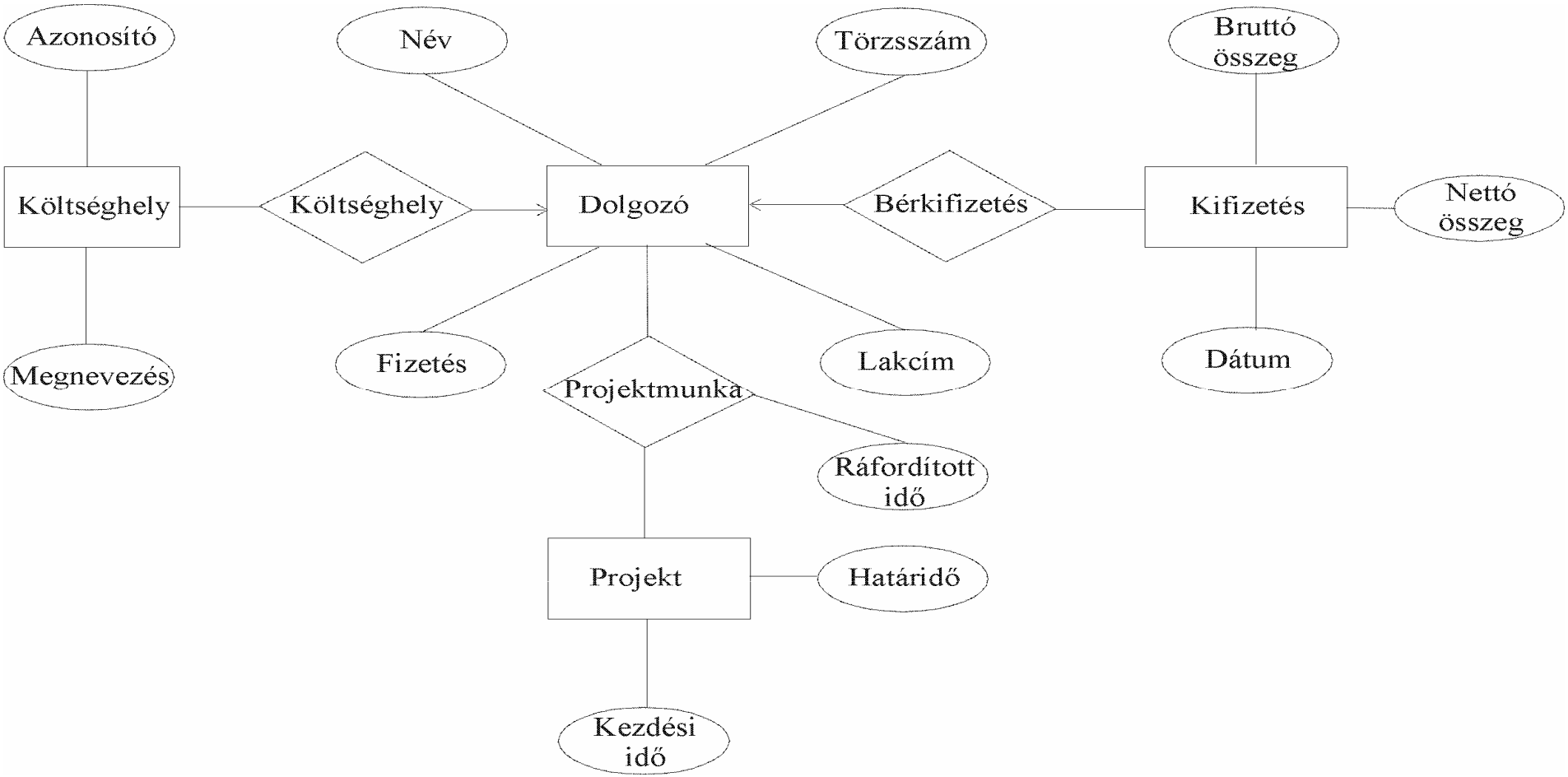
Egyed-kapcsolat diagrammok

Az adatmodell grafikus megközelítését szokás ***egyed-kapcsolat diagrammnak (ER=Entity Relationship diagramm)*** nevezni. Az egyed-kapcsolat diagramm elemei az adatmodell már ismert összetevői. Az egyes elemek grafikus reprezentációja a következőképpen készíthető el. Az **egyedeket téglalap** segítségével adjuk meg, amelybe beleírjuk az egyed nevét. Például a Dolgozó egyed ábrázolása a következő:

A **tulajdonságokat** hasonlóan ábrázoljuk, azzal a különbséggel, hogy itt téglalap helyett **ellipszist** használunk. Például a Dolgozó **egyed** Név tulajdonságának ábrája a következő:

A **kapcsolatok** esetében a **rombusz** jelet használjuk. Például a Dolgozó és a Kifizetés egyedek közötti kapcsolat a következő módon reprezentálható:

Példa: A már említett dolgozói nyilvántartás egyed-kapcsolat diagrammja a következőképpen nézhet ki:



Adatmodellek típusai

Az előző részben említett adatmodell elemekből különböző struktúrák igénybevételével alakítjuk ki az adatmodellt. Az adatbázis-kezelés fejlődése során három fontos adatmodellt alakítottak ki. Ezek a következők:

Hierarchikus modell

Hálós modell

Relációs modell

A **hierarchikus adatmodell**:

***Hierarchikus*.** Azért kapta ezt az elnevezést, mert az alapelve az, hogy az egyedeket a köztük lévő kapcsolat alapján hierarchiába rendezi. A hierarchikus modell leginkább egy-egy és egy-sok jellegű kapcsolatok megvalósítására használható. A kapcsolat alapján a két egyedtípus között hierarchiát definiálunk. Fontos, hogy a hierarchiában alul lévő egyedtípusnak csak egyetlen őse lehet. A hierarchiában felül lévő egyedtípushoz több, a hierarchiában lentebb lévő egyedtípus kapcsolódhat.

Az első hierarchikus ABKR az IBM által 1968-ban kifejlesztett ***Information Management System (IMS)*** volt. Mára a relációs modell már teljesen kiszorította a hierarchikust.

A hálós adatmodell

A ***hálós adatmodellt*** tekinthetjük a hierarchikus kiterjesztésének. A különbség az, hogy míg a hierarchikus modell gráfja csak fa lehet, a hálósnál tetszőleges gráf előfordulhat. Ez azt jelenti, hogy például egy egyedtípusnak több őse is lehet. A hálós adatmodellben a **sok-sok kapcsolatok** is kezelhetők, úgy hogy azokat két egy-sok kapcsolatra bontják fel. Magát a modellt, illetve a hozzá kapcsolódó adatbázis-kezelő nyelvet 1971-ben ismertette a **CODASYL DBTG (Conference on Data Systems and Languages Data Base Task Group).**

Relációs adatmodell

Ennek alapját a matematikából is jól ismert ***reláció*** jelenti.

A reláció tulajdonképpen **kétdimenziós táblázatos** adatelrendezést jelent.

Egy komplex **adatbázis-kezelő nyelv** került kifejlesztésre, amelyet **SQL-nek (Structured Query Language=Struktúrált Lekérdező Nyelv)** neveztek el.

Ennek segítségével nemcsak lekérdezéseket, hanem különböző adatbázis kezelési műveleteket is végrehajthatunk. Az SQL nyelvet beillesztve programozási nyelvekbe, fejlesztő környezetbe egyszerű, hatékony jól használható eszközt kapunk adatbázis-kezelési feladatok megoldására.

A relációs modellnek jól meghatározott elmélete van, amelynek alapjait **E. F. CODD** dolgozta ki még a 60-es évek végén, illetve a 70-es évek elején. Az elmélet bázisát a **halmazelméletből** veszi, magukat az adatokat tartalmazó táblákat halmazként definiálja, és így az adatokon végzett műveleteket is **halmazműveleteknek** felelteti meg.

A relációs adatmodell alapjai

- Minden egyes táblázat egyértelmű azonosítóval bír
- A sorok és oszlopok metszetében található adatok egyértékűek, vagy más szóval atomiak (azaz adatcsoportok, tömbök nem megengedettek). Ezeket szokásos elemi adatmezőnek is nevezni.
- Az oszlopokban található adatok azonos jellegűek, és egy előre definiált halmazból származnak
- Minden egyes oszlopnak egyedi elnevezése van
- A táblázat minden sorában ugyanannyi adat található
- A táblázatban nem lehet két azonos sor
- A sorok és az oszlopok sorrendje nem lényeges

A következő táblázat egy lehetséges példát mutat, amely a **Dolgozó** nevű táblázat egy részletét ábrázolja:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje
T234578	Kiss István	Eger	1968. 12. 11.
T456734	Nagy József	Budapest	1972. 01. 30.
T429877	Kovács János	Szeged	1967. 05. 12.

Látható, hogy a fenti táblázat teljesíti a felsorolt követelményeket, így tekinthető a relációs modell egy táblázatának.

Nézzük most, hogyan definiáljuk a ***reláció*** fogalmát.

Definíció: Legyenek $D_1,\ldots,D_n$ halmazok. Az $R \subseteq D_1 \times \ldots \times D_n$ halmazt **relációnak** nevezzük.

A táblázat sorait ***előfordulásoknak*** nevezzük.

Az előfordulások **rekordoknak** felelnek meg.

A reláció ***számosságát*** az összes előfordulás száma adja meg.

A reláció oszlopait ***attribútumoknak*** nevezzük.

Minden egyes attribútumhoz tartozik egy ***tartomány*****.** Ez az attribútumhoz tartozó D_i halmaz, amely azt mondja meg, hogy az adott oszlop milyen értékeket tartalmazhat. Szokás ezért a D_i halmazokat ***attribútumhalmaznak*** is nevezni. A reláció nevét és a reláció attribútumainak halmazát együtt szokás ***relációsémának*** nevezni.
Például a Dolgozó **reláció sémája** a következőképpen adható meg:

Dolgozó(A dolgozó törzsszáma, A dolgozó neve, A dolgozó születési helye, A dolgozó születési ideje)

Egy adatbázis több relációból áll.

Egy adatbázis relációsémáinak összességét ***relációs adatbázissémának*** nevezzük.

A reláció attribútumainak száma határozza meg a reláció **fokát.** A relációknak két típusát definiálhatjuk, amelyek adatbázis-kezelési szempontból lényegesek.

Az ***alap reláció*** olyan reláció, amely az adatbázisban a többi relációtól függetlenül létezik. A ***származtatott reláció*****,** vagy ***nézet*** más relációkból való konstruálással keletkezik. Például az adatbázisban tárolt **Dolgozó**

relációból származtathatunk egy új relációt **Dolgozónő** névvel, amely reláció csak a nő dolgozókat fogja tartalmazni.

Ahogy a fentiekből kitűnik, a relációs modell egyik fontos alaptulajdonsága, hogy a táblázat sorai és oszlopai metszetében található adatok **elemiek**.

Kapcsolatok a relációs modellben

A kapcsolatokat maguk a táblázatok tartalmazzák. Az alábbiakban ennek a megvalósítási technikáit mutatjuk be. Háromféle kapcsolatról beszéltünk.

Az **egy-egy, az egy-sok és a sok-sok** típusúakról. A relációs modellben mindegyiket különbözőképpen lehet megvalósítani.

Egy-egy kapcsolat. Ilyenkor csupán az szükséges, hogy a kapcsolatban részt vevő két egyed közül az egyiket reprezentáló tulajdonságok közül kiválasszunk **egy kulcsot.** Ezután a másik egyedhez tartozó táblázatban egy olyan attribútumot kell definiálnunk, amelynek lehetséges értékei az előzőleg kiválasztott kulcsmező értékeit vehetik fel.

Egy-sok kapcsolat esetében teljesen hasonlóan járhatunk el. Ekkor ugyanis ha annak az egyednek a szempontjából vizsgáljuk a kapcsolatot, amelynek előfordulásaihoz egy másik egyed előfordulás tartozik, pontosan ugyanazt a helyzetet kapjuk, mint az első esetben. Ebben az esetben tehát csak arra kell ügyelnünk, hogy a fenti technika egy irányban működik.

Példa

Nézzük most meg, hogyan nézhetnek ki a *Dolgozó* és a *Kifizetés* egyedek táblázatai, amennyiben a köztük lévő kapcsolatot is reprezentáljuk:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje
T234578	Kiss István	Eger	1968. 12. 11.
T456734	Nagy József	Budapest	1972. 01. 30.
T429877	Kovács János	Szeged	1967. 05. 12.

A kifizetés azonosítója	A kifizetés dátuma	A kifizetett nettó összeg	A dolgozó törzsszáma
K23756	1999. 10. 01.	120000	T234578
K23757	1999. 11. 01.	89000	T234578
K23758	1999. 10. 01.	167000	T429877

A legbonyolultabb kapcsolatfajta a **sok-sok** kapcsolat. Sajnos ezt már nem lehet olyan egyszerű módon megadni, mint az egy-egy és egy-sok kapcsolatokat. Mivel mindkét egyed irányából nézve több előfordulás tartozhat egy előforduláshoz, egyetlen lehetőség, hogy az egymáshoz kapcsolódó előfordulásokat külön összepárosítjuk. Erre alkalmas lehet **egy speciális reláció**, vagy más néven **kapcsolótábla,** amely két attribútumból áll. Az egyik attribútum az egyik egyed, a másik a másik egyed egy kulcsának lehetséges értékeit veheti fel értékül. Az összetartozó előfordulásokat a hozzájuk tartozó kulcsértékeknek a kapcsolótáblában való összepárosításával adjuk meg.

A téma azonosítója	A munka kezdete	Befejezési határidő	Témavezető neve
T25	1998. 06. 01.	1999. 12. 31.	Nagy Ádám
T26	1999. 01. 01.	2000. 06. 01.	Mekk Elek
T27	1999. 03. 01.	2002. 12. 31.	Remek Jenő

A téma azonosítója	A dolgozó törzsszáma
T25	T234578
T25	T456734
T25	T429877
T26	T456734
T26	T429877
T27	T456734

Funkcionális függések és jellemzésük

A relációs modell egyik legfontosabb fogalma a funkcionális függés. Segítségével a táblázat attribútumai között bizonyos összefüggéseket állapíthatunk meg. Vagyis **egy funkcionális függés megadja azt, ha az adott relációban valamely attribútumokon vett értékek meghatározzák más attribútumok értékeit.** Segítségével ugyanis csökkenthető a modell redundanciája, ugyanis a funkcionális függésben lévő attribútumokat külön relációkba szervezhetjük, anélkül hogy ezzel bármilyen információt elveszítenénk.

Definíció: Legyen R a $D_1,...,D_n$ halmazokon értelmezett reláció. Legyen továbbá $A,B \subset \{D_1,...,D_n\}$ két attribútumhalmaz. Azt mondjuk hogy B **funkcionálisan függ** A **-tól** az R relációban ($A \to B$), ha bármely két $r1,r2 \in R$ **-re** $r1(D_i) = r2(D_i), \forall D_i \in A$ esetén teljesül, akkor ebből következik hogy $r1(D_j) = r2(D_j), \forall D_j \in B$ **-re**.

Példa

Tekintsük most ismét a *Dolgozótéma* **relációt**, és tegyük fel, hogy ezúttal a következő módon terveztük meg

A dolgozó törzsszáma	Téma azonosítója	Témavezető neve
T234578	T25	Nagy Ádám
T456734	T25	Nagy Ádám
T429877	T25	Nagy Ádám
T456734	T26	Mekk Elek
T429877	T26	Mekk Elek
T429877	T27	Remek Jenő

Láthatjuk, hogy ha a *Téma azonosítója* oszlop értékei megegyeznek, akkor a *Témavezető* oszlop értékei is megegyeznek. Így a fenti definíció alapján teljesül a *Téma azonosítója -> Témavezető* **funkcionális függés**.

Azt is könnyen felfedezhetjük, hogy a függés egyfajta redundanciát jelent, ugyanis felesleges háromszor leírni, hogy a *T25* azonosítójú projekt vezetője *Nagy Ádám*.

Az első fontos kérdés, - ami gyakorlati szempontból is jelentős - hogy ha ismerünk bizonyos függőségeket, akkor ezekből kikövetkeztethetünk-e újabb függőségeket. Amennyiben igen, akkor a következtetési szabály alkalmazásával egy függőséghalmazból újabb függőségeket vezethetünk le.

Armstrong-axiómák. Ezek segítségével már meglévő függőségekből új függőségek származtathatók.

Az egyik legegyszerűbb következtetési szabály azt mondja ki, hogy ha az $A, B, C \subseteq R$ attribútumhalmazokra teljesülnek az $A \to B$ és $B \to C$ függőségek, akkor teljesül az $A \to C$ függőség is. Ezt a tulajdonságot nevezzük **tranzitivitásnak.**

Nyilvánvalóan igaz az a szabály is, hogy a B halmaz minden B' részhalmazára is teljesül az $B \to B'$ függőség. Ezt nevezzük ***triviális függőségnek***, vagy **reflexivitásnak.**

A következő két szabály szorosan kapcsolódik egymáshoz, mivel lényegében egymás ellentettjei.

Az első az **egyesítési szabály,** amely azt mondja ki, hogy ha teljesül egy $A \to B$ és egy $A \to C$ függőség, akkor teljesül az $A \to B \cup C$ függőség is. Szavakkal kifejezve azonos bal oldalú függőségek jobb oldalán szereplő attribútumhalmazait egyesíthetjük.

Ugyanakkor a funkcionális függőség jobb oldalán szereplő attribútumhalmazra teljesül az, hogy annak minden részhalmaza is függ a baloldaltól. Vagyis ha $A \to B$ és $C \subseteq B$, akkor $A \to C$ is teljesül. Ezt nevezik **szétvághatósági szabálynak.**

Ugyancsak fontos a **bővítési szabály.** Ez azt mondja ki, hogy ha egy funkcionális függés mindkét oldalát ugyanazzal az attribútumhalmazzal bővítjük, akkor a függés továbbra is megmarad a két halmaz között.

Formálisan ha $A \to B$ és C tetszőleges attribútumhalmaz, akkor $A \cup C \to B \cup C$.

Kulcsok a relációs modellben

A relációs modellben külön definíciót adtak a kulcsra. Ez tulajdonképpen megfelel a korábban már ismertetett kulcs fogalomnak, csak a relációs modellben pontos matematikai jellegű definíció is adható.

Definíció: Legyen $A = \{A_1, \ldots, A_n\}$ az R reláció attribútumhalmaza. A $K \subseteq A$ halmazt a **reláció kulcsának** nevezzük, ha

(1) a K attribútumain felvett értékek egyértelműen meghatározzák a reláció elemeit, azaz $K \rightarrow A$, valamint

(2) nincs K-nak olyan valódi részhalmaza amelyre ugyanez teljesül, azaz K **minimális.**

Az olyan attribútumhalmazokat,a melyekre csak az (1) tulajdonság teljesül **szuperkulcsnak** nevezzük.

Meg szoktuk különböztetni azt az esetet, amikor a kulcs egy attribútumból áll. Az ilyen jellegű kulcsokat **egyszerű kulcsoknak** nevezzük.

Ellenkező esetben **összetett kulcsokról** beszélünk.

Egy relációnak természetesen több kulcsa is lehet, és minden egyes attribútumról eldönthető hogy tartozik-e kulcshoz vagy sem.

Elsődleges attribútumoknak nevezzük azokat az attribútumokat, amelyek valamely kulcshoz tartoznak,

másodlagosaknak azokat, amelyekre ez nem teljesül.

A relációs modell elméletében a kulcsok között nem szokás különbséget tenni, a gyakorlatban azonban gyakran szükséges hogy meghatározzunk egy **elsődleges kulcsot**, amely főleg az ABKR számára fontos a rekordok fizikai elhelyezéséhez.

A definíció alapján a kulcshoz tartozó attribútumokon **egy értéksorozat csak a reláció egyetlen sorában fordulhat elő**.

Ugyanis mivel a kulcs a teljes attribútum halmazon felvett értékeket meghatározza, ezért ha lenne két olyan sor, amelyekben a kulcshoz

tartozó attribútumokon azonos értékek vannak, akkor két azonos sornak kellene lenni a relációban. Ez pedig nem lehetséges.

Az olyan attribútumokat, amelyek egy másik relációban alkotnak kulcsot **külső kulcsnak, vagy idegen kulcsnak** szokás nevezni. A külső kulcsoknak a **kapcsolatoknál** van jelentősége.

Külső kulcsok különböző formában jelenhetnek meg a relációs modellben. Előfordulhat az, hogy **egy kulcs saját táblájában szerepel külső kulcsként**. Például elképzelhető hogy a bérszámfejtő rendszerben tárolni kívánjuk hogy az egyes dolgozóknak ki a főnöke. Ebben az esetben a *Dolgozó* tábla a következő lehet:

Dolgozó(A dolgozó törzsszáma, A dolgozó neve, A dolgozó születési helye, A dolgozó születési ideje, A vezető törzsszáma)

A vezető törzsszáma attribútum tulajdonképpen külső kulcs, hiszen adatait *a A dolgozó törzsszáma* kulcs értékei közül veszi fel, azonban az a speciális helyzet áll elő, hogy a külső kulcs éppen ugyanahhoz a táblázathoz tartozó kulcs. Ez egyfajta **rekurzív kapcsolatot** jelent. Az is előfordulhat, hogy egy táblában egy külső kulcs többféle szerepben található meg.

Relációk normalizálása

A tervezés során olyan adatstruktúrákat alakítsunk ki, amelyek segítik a hatékony adatkezelést. **Fontos hogy egy-egy táblába csak a valóban logikailag összetartozó adatok kerüljenek, és hogy minél kevesebb ismétlődés legyen az adatok között.**

A relációs modellben külön eljárást fejlesztettek ki arra vonatkozóan, hogy az adatok megfelelő strukturálását, a redundancia csökkentését elősegítsék.

Ez a módszer a ***normalizálás*.**

Bevezettek ***első, második, harmadik, Boyce-Codd, negyedik, ötödik*** **normálformát** is. Ezek közül a legnagyobb jelentősége a **harmadik és a Boyce-Codd** normálformáknak van.

A legfontosabb műveletek a **beszúrás, a módosítás és a törlés.** Nézzük, milyen problémák merülhetnek fel ezeknél.

Adatkezelési műveletek anomáliái

Beszúrási anomáliáról beszélünk abban az esetben, amikor egy adatrekord beszúrása egy másik, hozzá logikailag nem kapcsolódó adatcsoport beszúrását kívánja meg.

A következő problémát a már felvitt **adatok módosítása** okozhatja. Abban az esetben, ha egy relációban egy adat módosítása több helyen történő módosítást igényel, akkor ***módosítási anomáliáról*** beszélünk.

A harmadik anomália az **adatok törlésekor** fordulhat elő. Amennyiben egy adat törlésével
másik, hozzá logikailag nem kapcsolódó adatcsoportot is elveszítünk, **törlési anomáliáról** beszélünk.

Az anomáliákat mindig az okozta, hogy logikailag nem összetartozó adatokat szerveztünk egy struktúrába.

Nagyon lényeges, hogy a tervezés során ezt a hibát ne kövessük el. Ezért **el kell választanunk egymástól a logikailag összetartozó, vagy mondhatjuk úgy hogy összefüggő adatokat.** S ebben lesz fontos szerepe a funkcionális függésnek, ugyanis ennek segítségével tudjuk

kiválogatni hogy mik azok az attribútumok, amelyek ilyen csoportokat alkotnak. Ezeket aztán különválasztjuk, **önálló táblákat** alkotva. Így csökkenthetjük, illetve szüntethetjük meg a fenti problémákat.

Normálformák

A relációk felbontásával különböző normálformájú relációkat kapunk.

A legegyszerűbb az **első normálforma.**

Azt mondjuk, hogy **egy reláció első normálformában van,** ha a relációban minden érték elemi, vagyis a reláció nem tartalmaz adatcsoportot.

Ezt ugyanis már a relációs modellbeli táblázat alapvető tulajdonságai között is megköveteltük. Ha az alaphalmazok csak elemi adatokat tartalmaznak, akkor a megadott reláció rögtön első normálformájú is lesz.

Csoportos adatok esetén azonban lényeges, hogy a keletkezett táblázatot első normálformájúvá alakítsuk. Tekintsük a következő táblázatot:

A dolgozó törzsszáma	A dolgozó neve	A dolgozó születési helye	A dolgozó születési ideje	Szakképzettség
T234578	Kiss István	Eger	1968.12.11.	Közgazdász
T456734	Nagy József	Budapest	1972.01.30.	Programozó matematikus, közgazdász
T429877	Kovács János	Szeged	1967.05.12.	Programtervező matematikus, számítástechnika tanár

Ezt a táblázatot az alább módon alakíthatjuk át első normálformájú relációvá:

A dolgozó törzsszáma	**A dolgozó neve**	**A dolgozó születési helye**	**A dolgozó születési ideje**	**Szakképzettség**
T234578	Kiss István	Eger	1968.12.11.	Közgazdász
T456734	Nagy József	Budapest	1972.01.30.	Programozó matematikus
T456734	Nagy József	Budapest	1972.01.30.	Közgazdász
T429877	Kovács János	Szeged	1967.05.12.	Programtervező matematikus
T429877	Kovács János	Szeged	1967.05.12.	Számítástechnika tanár

Egy **relációt második normálformájúnak** szokás nevezni, ha első normálformájú, és egyetlen másodlagos attribútuma sem függ egyetlen kulcsának valódi részhalmazától sem.

A **második normálforma** azt jelenti, hogy a relációban nincsenek a kulcs részeitől való nem triviális függőségek.

Ez a normálforma tulajdonképpen egy közbenső lépés a harmadik normálforma felé. Ezért jelentősége csupán annyi, hogy a normalizálási tevékenységet kisebb lépések beiktatásával segíti, vagyis ezáltal a normalizálást végző személynek először csak a kulcsokat és azok részhalmazait kell megvizsgálni, hogy a nem kívánatos függőségeket eltávolítsa a relációból.

A második, harmadik és Boyce-Codd normálformák esetén a normalizálási technikák megegyeznek.

Gyakorlati szempontból a leglényegesebb normálforma a harmadik.

Erre vonatkozóan az irodalomban több definíció található, amelyek egymással ekvivalensek.

Definíció: Egy *R* reláció ***harmadik normálformában*** van, ha

1) második normálformájú és

2) egyetlen másodlagos attribútuma sem függ tranzitíven kulcstól.

Egy $A \to C$ funkcionális függést *tranzitív függésnek* nevezünk, ha létezik olyan *B* attribútum halmaz, hogy $A \to B$ továbbá $B \to C$.

Vizsgáljuk meg, mit is jelent ez a definíció.

Az hogy egy másodlagos attribútum nem függ tranzitíven egy kulcstól azt jelenti, hogy nem találhatók olyan attribútumok, amelyektől funkcionálisan függ bármely másodlagos attribútum.

Ugyanis ha ilyen lenne, akkor lenne tranzitív funkcionális függés, mivel a kulcstól minden más attribútum függ.

Ez alapján a harmadik normálforma másik definíciója a következő:

Definíció: Egy *R* reláció ***harmadik normálformában*** van, ha második normálformájú és a másodlagos attribútumok között nincsen funkcionális függés.

Végezetül lehet adni egy olyan definíciót is, amely nem igényli a második normálforma fogalmát. Azon szemléletmód esetén, amikor egyből a harmadik normálforma definíciójával indulnak, ezt szokás használni.

Definiáljuk a **Boyce-Codd normálformát**, amely a harmadik normálformának egy további általánosítása.

Definíció: Egy R reláció akkor és csak akkor van ***Boyce-Codd normálformában*****,** amennyiben ha létezik az $A \to B$ függőség valamely A, B attribútum halmazok esetén, akkor az A az R szuperkulcsa.

A szuperkulcs definíciója alapján a Boyce-Codd normálforma lényegében azt jelenti, hogy

minden nem triviális funkcionális függőség bal oldalának tartalmaznia kell egy kulcsot.

Mint láttuk a második, harmadik és Boyce-Codd normálformák teljesülése mindig azon múlik, hogy a relációban vannak-e bizonyos jellegű funkcionális függőségek.

Ha szeretnénk elérni azt, hogy egy adott normálforma kritériuma teljesüljön, akkor **el kell tüntetnünk a számunkra nem kedvező függőségeket a relációból.**

Ez egy **felbontással történhet, amelynek során a relációt két önálló relációra bontjuk.**