SELYE JÁNOS EGYETEM
GAZDASÁGTUDOMÁNYI KAR

Dr. Kiss Gábor

ADATBÁZIS-KEZELÉSI ALAPISMERETEK

2021

Dr. Kiss Gábor

Adatbázis-kezelési alapismeretek

Tankönyv

Szerző: Dr. Kiss Gábor
Selye János Egyetem, Gazdaságtudományi és Informatikai Kar

ISBN: 978-80-8122-389-1

Előszó

Ha valaki informatikával kezd el foglalkozni, idővel találkozik a nagy mennyiségű adatok megfelelő tárolásának és hatékony visszakereshetőségének problémájával.

A programfejlesztések során nem csak valamilyen számítási műveletet kell elvégeznünk, egyre inkább eltolódik abba az irányba, hogy adatokat tároljunk, keressünk vissza, végezzünk el rajtunk valamilyen műveletet.

Ugyan egy sima szöveg fájl használata is megfelelőnek tűnhet, mégis a fájl méretének a növekedésével az abban tárolt adatok elérési/megtalálási ideje egyre jobban növekszik és már egyre több időt kell várnunk, hogy a kérésünkre pontos választ kapjuk a programtól, hiszen sorról-sorra kell végig néznünk a letárolt adatokat, megegyeznek-e az általunk keresettel.

Emellett nem egyszerű leprogramozni egy szövegfájlból történő adattörlést sem.
A típusos fájlok kezelési lehetősége az egyes programozási nyelvekben már lehetővé tette az adott rekordra történő pozícionálást, a törlés megvalósítása is megjelent az utolsó adatnak a törlendő adat helyére történő másolásával, majd az utolsó törlésével, a keresés továbbra is szekvenciális maradt.

Az adatbázis-kezelő rendszerek az adatok strukturált tárolására és gyors elérésére kínálnak megoldást, de ahhoz, hogy hatékonyan tudjuk őket használni, néhány dologgal tisztában kell lennünk.

Ma már nem csak az adattárolás és hatékony visszakeresés a cél, hanem a tárolt adatok alapján gépi intelligencia tanítása is megjelenik igényként, ezért még fontosabb a körültekintő tervezés az adatbázisok létrehozásánál.

A megfelelő adatbázistervezés támogatására készült ez a tankönyv, alapul véve Hector Garcia-Molina and Jeffrey D. Ullman and Jennifer Widom - Database Systems: The Complete Book 2002-ben megjelent könyvét.

Emellett Dr. Bércesné Dr. Novák Ágnes kolléganőmmel közösen tartott egyetemi órák tananyaga, jegyzete és tapasztalata bővíti a tankönyvet.

A definíciók, magyarázatok jellemzően matematikai alapon nyugszanak, mégse ijedjen meg senki, általánosabb megfogalmazás is segíti a könnyű megértést.

A tankönyv nem helyettesíti az órán való részvételt, a tananyag sikeres elsajátításának támogatása a célja a rendszeres óralátogatás mellett.

Tartalomjegyzék

I. Bevezetés

A tankönyv célja, megfelelő elméleti ismereteket nyújtani az adatbáziskezelés területén, kitérve a legfontosabb lépésekre, melyeken keresztül egy adott feladat megoldható úgy, hogy az adataink optimálisan kerültnek tárolásra, a redundanciát megfelelő mértékben lecsökkentettük és az adatok közötti kapcsolat biztosítja, hogy ne veszítsünk információt.

A könyvben számtalan példát mutatunk be, így segítve az elméleti ismeretek megértését és gyakorlatba történő átültetését.

Előfordulhat, hogy az életünkben adódó probléma megoldására a tankönyvben lévő ismeretek alapján tudnánk optimális megoldást adni, mégis a törvényi szabályozásnak való megfelelés miatt egy nem optimális adatbázis szerkezettel kell megelégednünk. Erre is mutatunk be példát a tankönyvben.

A könyv által átadott ismeretek alapján mindenki képes lesz egy meglévő adatbázis szerkezetét megérteni, észrevenni az összefüggéseket és szükség szerint módosításokat elvégezni rajta.
Az igazi kihívás viszont akkor jön el, amikor nekünk kell egy teljesen új adatbázis szerkezetét megtervezni és azt kezelni, de a könyv alapján erre és képesek lesznek a tisztelt olvasók.

II. Alapok

Ebben a fejezetben több olyan témakör megjelenik, melyek szükségesek lesznek a könyv későbbi részének olvasása során a könnyebb megértéshez, illetve az olvasóban az alapismeretek után már körvonalazódni fog, mivel is foglalkozik igazán az adatbázis-kezelés.

Alapfogalmak

Az adat fogalma

Az adat észlelt, de nem értelmezett, jelentés nélküli fogalom. Objektív tények összessége, egy nyersanyag, mely az eseménynek csak egy részét írja le. Pl. számla tartalma nem mond semmit arról, miért megy oda valaki vásárolni, visszatér-e, stb. A lényeges adatok azonosítását a túl sok adat megnehezíti.

Az információ fogalma

Az információ az értelmezett adat, jelentéssel bíró fogalom. Csökkenti a bizonytalanságot csökkenti, viszont szubjektív, személytelen, általános üzenet.
A szemléletmódon változtat. Áramlása formális és informális csatornákon is történik.

Az adatbázis fogalma

Nagy mennyiségű információ olyan struktúrált együttese, melyet adatbázis-kezelő rendszeren keresztül lehet elérni és **hatékonyan** kezelni.
Az adatbázis-kezelő rendszert tulajdonságain keresztül definiáljuk. adatbázis-kezelő rendszer re vonatkozó elvárások:

Adatbáziskezelő rendszerek és a belőlük kifejlődőtt rendszerek története

Az **adatbáziskezelő rendszerek** az 1960-as években jelentek meg az Egyesült Államokban [1].

Cél:
Tranzakciók, műveletek végrehajtása egy vagy több erőforráson. Írás, olvasás, törlés, módosítás az adatbázisban. A szervezet rutinszerű, ismétlődő, alapvető funkcióit dolgozza fel

Tovább fejlődve és bővített funkciótartalommal a 70-es években megjelentek az **üzleti információsrendszerek**.

Cél:
A szervezet számára információt nyújt, tervező, szervező, kontrolling tevékenységeket valósít meg, mellyel a múlt, a jelen, a jövőbeli helyzetről tudunk meg információt. Feladatai között megjelenik a jelentésgenerálás, online ellenőrzés, valós idejű lekérdezések.

Az 1980-as években megjelennek a **döntéstámogató rendszerek** (Decision Support Systems).

Cél:
Felhasználóbarát kezelői felületen megvalósítja a szimulációs modellezést, online valós idejű tervezést a rendelkezésre álló nagy mennyiségű adat alapján. Strukturált döntéshozatal automatizálása válik lehetővé, de nem helyettesítik a döntéshozót. Felhasználási területek: pénzügyi tervező rendszerek, piackutatás, előrejelzések, vállalati elemzések.

Az 1980-as években a döntéstámogató rendszerek tovább fejlődéseként és a mesterséges intelligencia kutatások eredményeként megjelennek a **tudásbázisú rendszerek**.
A problémamegoldó gondolkodás makroszinten való modellezése valósul meg bennük, a gondolkodási folyamatot egészben tekintik és a szakértők problémamegoldó gondolkodását követik. Heurisztikus ismeretek felhasználásával vonják le a következtetéseket, melyek után javaslatot tesznek a megoldás kidolgozására. A problémamegoldásuk minősége függ a tudásbázis tartalmától.

Szintén az 1980-as években jelennek meg a **szakértő rendszerek**. Különbség a tudásbázisú rendszerekhez képest, hogy egy szűk problématerület kezelésében nyújtanak magasszintű teljesítményt (pl. orvosi kutatások)

Az 1990-es években a **felsővezetői információrendszerek** bukkanak fel, melyekben új adattárolási és kezelési megoldás jelenik meg. Lehetőség van részletek megtekintésére a lekérdezések esetén pedig rugalmasság megvalósíthatóságára, többszempontú elemzésre.

A fentiek közös jellemzője, hogy megfelelő működésükhöz egy jól megtervezett adatbázisra és benne releváns adatokra van szükség.

Elvárások egy adatbáziskezelő rendszerrel szemben:
1.
- Új adatbázis létrehozásának lehetősége
- Adatok logikai szerkezetének leírásának lehetősége
- Adatdefiníciós Nyelv Data Definition Language = DDL használatának lehetősége

2.
- Adatok hatékony lekérdezése: Adatmanipulációs nyelv Data Manipulation Language (DML)

3.
- Biztonságos adattárolás:
- jogosulatlan felhasználók elleni védelem
- meghibásodások elleni védelem adatbázis-kezelő rendszer helyreállítási technikák által

4.
- Több felhasználó egyidejű hozzáférése, konkurrenciakezelés, tranzakciókezelés

Történeti áttekintés (témák szerint)

Példa:

Banki rendszerek : Felhasználói programok:
* egy számlára betenni-kivenni
* új számlát nyitni
* egyenleget számotarni
* havielszámolásokat írni
Adatelemek:
ügyfél(név, cím,számlaszám)
számla(számlaszám, egyenleg, típus)

Példa:

Repülőgép helyfoglalási rendszerek
Adatelemek:
vevő(név, cím, telefon, járatszám)
járat(járatszám, ind., érk., reptér..) ülőhely(járatszám, ülőhelyaz.,vevő)

Példa:

Vállalati nyilvántartások
- Eladások, kimenő számlák, bejövő számlák - Mik lehetnek az adatelemek?

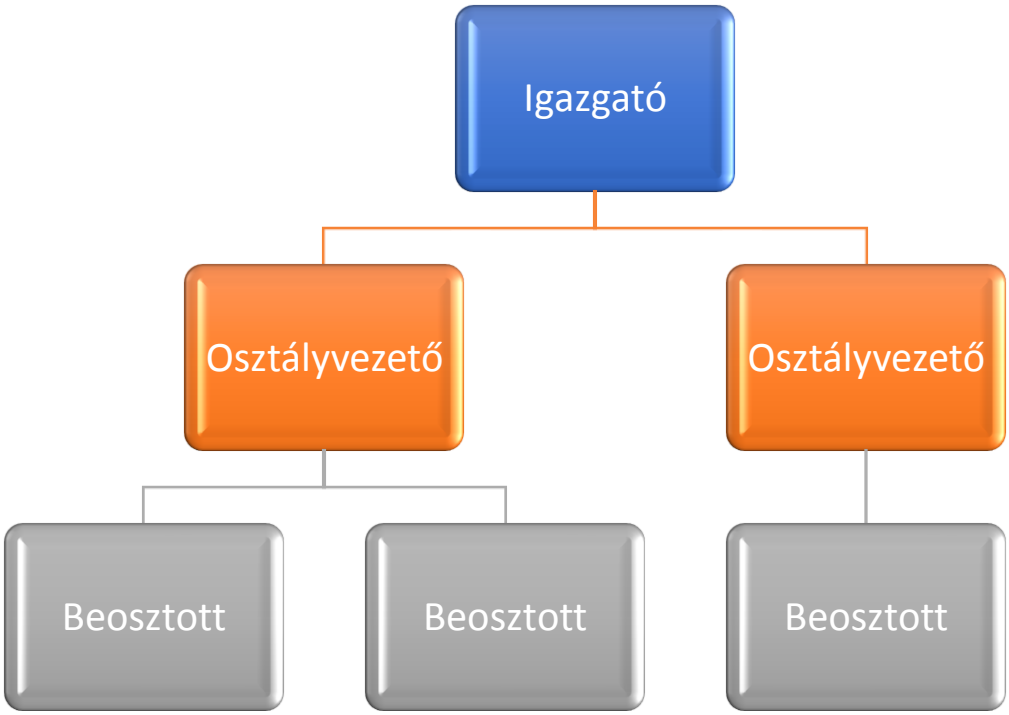
Fájlkezelő rendszerek

- Fájlok, programok különböző szerkezetűek
- Adatfeldolgozás nehézkessége-új kérdések megválaszolása rendszerint új programokat igényel
- Az új programok írása is nehéz, hiszen az adatok különböző fájlokban lehetnek
- Hivatkozási épség nehéz ellenőrzése
- Atomosság fájlszintű
- Konkurrencia-többfelhasználóegyidejűhozzáférésének kezelése
- Biztonsági kérdések-fájlszintű hozzáférés
- Absztrakciós szint alacsony (fizikai szint ismerete szükséges a felhasználói programok megírásához)

Történeti áttekintés (adatmodell szerint)

Első adatbázis-kezelő rendszerek adatmodell szükségessége:

- hierarchikus (fa) adatmodell (1. ábra)



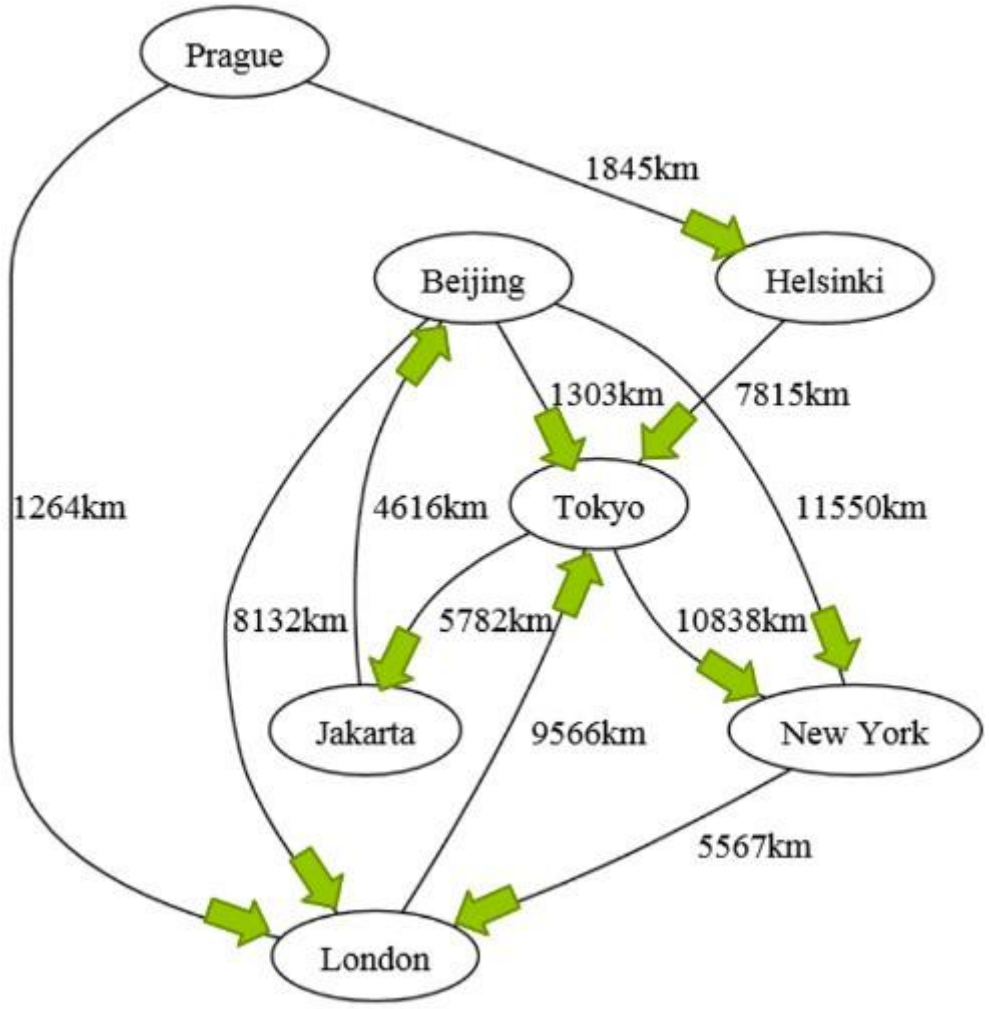
1. ábra Egy vállalat hierarchikus felépítését leírós adatmodell szerkezete

A hálós adatmodell esetén könnyen lehet leírni a hierarchikus kapcsolatokat az egyes adatok között [2]. Az Egyesült Államokban az 1960-as évek elején ez volt az uralkodó irány az adatok tárolására. Ha megnézzük az 1. ábrát, láthatjuk, hogy egy vállalati hierarchia jó leírható vele. Lekövethetjük, hogy melyik beosztott, melyik osztályvezetőhöz tartozik. Könnyen tudunk a Fa szerkezettel jól leírható adatmodellbe új adatot, pl. beosztottat beilleszteni.
A fa szerkezetet már mindenki ismeri, elég csak a számítógépen az alkönyvtárak/mappák felépítésére gondolni.

Az adatmódosítás, pl. egyik beosztottnak egy másik osztályvezető alá történő átszervezése már bonyolultabb eset, át kell „linkelni” máshoz, mely sok esetben csak két lépésben valósítható meg (új elemként felvinni, majd a régi helyről törölni). Egy adott elemhez csak egy útvonal létezik.

Amennyiben a hierarchia egy adott összetettséget meghalad (túl sok szintje lesz és szintenként túl sok elem), már nem hatékony az adatkeresés, adatmódosítás.

- hálós (gráf) adatmodell (2. ábra)



2. ábra Egy légitársaság útvonalrészlete

A hálós Adatmodell esetén nem hierarchikus a kapcsolódás az elemek között, hanem egyenrangú. Bármelyik irányban bővíthető, új elemmel egy másik elemhez csatlakoztatva, pl. a 2. ábrán a New York-i elemhez kapcsolni LosAngelest. Egy adott elem eléréséhez több útvonal is létezhet, pl. útvonaltervező szoftverek. A gráf szerkezet is ismert pl. matematikából már mindenki számára.
A könnyű bővíthetőség és több létező útvonal esetén az adatok hatékony elérése előnyként jelenik meg a hierarchikus adatmodellhez képest, az 1960-as évek második felében kezdett jobban elterjedni a használata. Megfelelő összetettség esetén viszont az adatok megtalálása már hatékonysági problémát jelentett

- relációs adatmodell (Codd, 1970)

1. táblázat Relációs adatmodell

Személyi azonosító	Név	Beosztás	Fizetés
AD341122	Kovács Béla	Osztályvezető	3500 Euro
GH782325	Nagy Attila	Igazgató	6700 Euro
OI561267	Balázs Csilla	Adminisztátor	2150 Euro

1970-ben egy angol matematikus, informatikus Edgar Frank Ted Codd az IBM kutatójaként létrehozta az adatbázis-kezelő rendszerek részére a relációs adatmodellt (1. táblázat), mely könnyen elképzelhető egy táblázatként, melynek van egy fejléc sora, alatta pedig a letárolt adatok [3]. A mai napig a legszélesebb körben elterjedt adatmodell annak ellenére, hogy már azóta az objektum orientált adatmodell is kidolgozásra került az objektum orientált programozás térhódításával, ugyanis hatékonyságában a relációs adatmodell felülmúlja.
Nevét onnan kapta, hogy az adatok közötti kapcsolat a kialakítás alapja.
A 1. táblázatban a személyi azonosító, melynek értéke meg tudja határozni a név, a beosztás és a fizetés értékét, de elképzelhető, hogy maga a beosztás is meghatározza adott esetben a fizetést. Ezen kapcsolatoknak az átgondolása és kiderítése a relációs adatbázis létrehozásának az egyik legfontosabb része (lásd később).

Alapfogalmak a relációs adatbázis-kezelésben:

Reláció/tábla: az az adathalmaz, mely az adatbázisban a logikailag összetartozó adatokat tartalmazza (pl. az 1. táblázatban szereplő adatok)
Attribútum/tulajdonság/mező/oszlop: minden egyed egy tulajdonsága (pl. az 1. táblázat alapján mindenkinek tudjuk a fizetését)
Rekord/sor: egy egyed minden tulajdonsága (pl. az 1. táblázat alapján egy személyről tudjuk minden tárolt adatát)
Funkcionális függőség: B attribútum értéke funkcionálisan függ A attribútum értékétől ($A \rightarrow B$), ha minden olyan sorban, ahol A attribútum értéke azonos, B attribútum értékének is azonosnak kell lennie (pl. az 1. táblázat alapján, ha lenne két olyan sor, ahol a személyi azonosító értéke azonos, azonosnak kellene lennie a névnek és a többi adatnak is, tehát személyi azonosító értéke meghatározza a név értékét).
Kulcs jelölt: az az attribútum halmaz, melynek értéke meghatározza a reláció egy sorát (pl. az 1. táblázat alapján a személyi azonosító, melynek értéke minden több adatot meghatároz).
Kulcs: a kulcsjelöltek közül a minimális (a legkevesebb attribútumból álló)
Idegen kulcs: az az attribútum halmaz, amely egy másik táblában kulcs.
Adatbázis: egy adott feladat megoldásához tárolandó adatok (szükség esetén külön táblákba szervezve) és a közöttük lévő kapcsolatok halmaza.

Széleskörű elterjedtsége és hatékonysága miatt a könyvben a relációs adatbáziskezelésre fókuszálunk.

Az adatbázis-kezelő rendszer-t használó személyek az alábbi feladatokkal, illetve jogosultságokkal vannak ellátva a sikeres megvalósítás és felhasználás érdekében:

Adatbázis-kezelő rendszer - Felelős személy:
Adatbázis adminisztrátor (rendszergazda) feladatai:
- sémamódosítás (adatbázis tábláinak szerkezetének módosítása)
- sémadefiniálás (adatbázis tábláinak szerkezetének kialakítása)
- fizikai szervezés módosítása
- megszorítások megfogalmazása (pl. a számla dátuma automatikusan legyen kitöltve, nem a felhasználó által)
- megszorítások módosítása
- hozzáférési jogok biztosítása (kinek van joga új adatot felvinni, kinek van joga módosítani, stb.)

Adatbázis-kezelő rendszer - Felhasználók:
- adatbázis-kezelő rendszer adminisztrátor (rendszergazda)
- felhasználói programok írói
- nem szakember felhasználók (pl. adatrögzítők)

Adatbázis-kezelő rendszer felépítése

Séma definiciós rész

- táblaszerkezetek kialakítása

Lekérdezés feldolgozó (program):

- DDL (Data Definiton Language, Adatdefiníciós nyelv) / DML (Data Manipulation Language, Adatmanipulációs nyelv) interpreter / compiler
- DML előfordító
- optimalizálók:
    - kérdések optimalizálása.
    - fizikai elérés optimalizálása

Tárkezelő (program):

- megszorítások ellenőrzése
- fájlkezelő (file-manager):
    fájlok tényleges elhelyez(ked)ése
- pufferkezelő:
    blokkok mozgatása

Tranzakciókezelő (program):

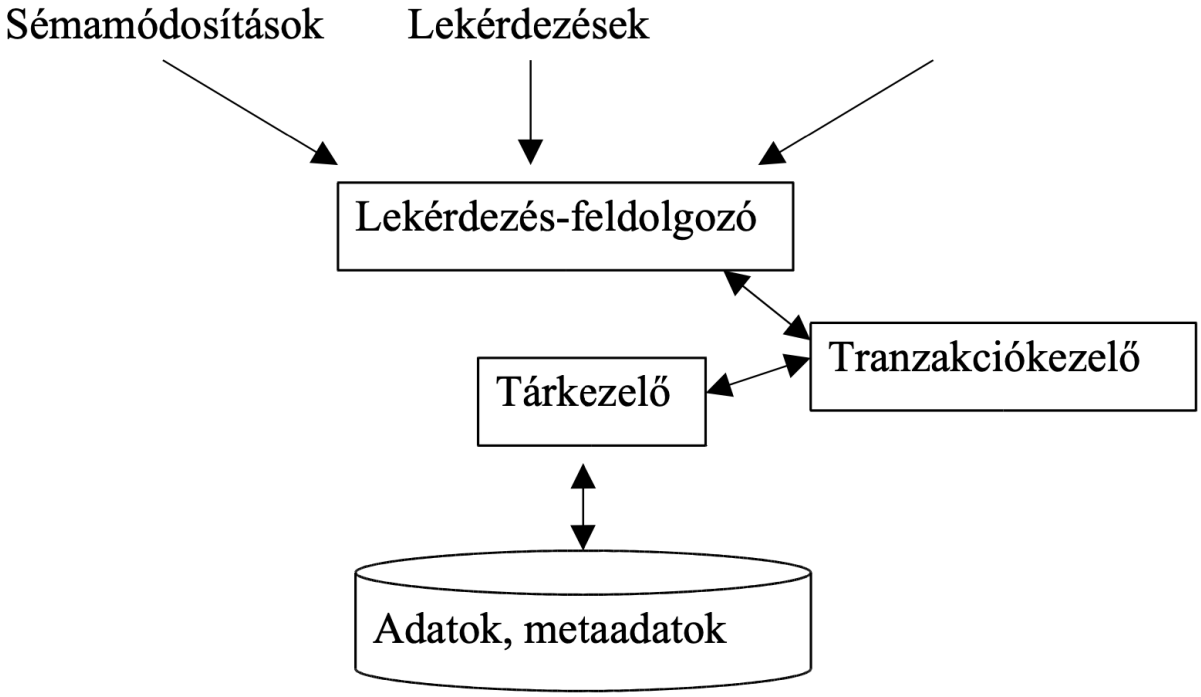
Ügyel a tranzakciók helyes kivitelezésére:
- Atomosság: mindent vagy semmit
- Következetesség
- Elkülönítés

Eszközei:
- zárolás,érvényesítés
- Tartósság (naplózás, tükrözés)

Fizikai tároló (eszköz):
- a tényleges adatok (szűkebb értelemben vett adatbázis)
- metaadatok (adatok típusa, hossza, megszorítások):
- indexek
- statisztikai adatok
- adatszótárak : adatszerkezetek leírása

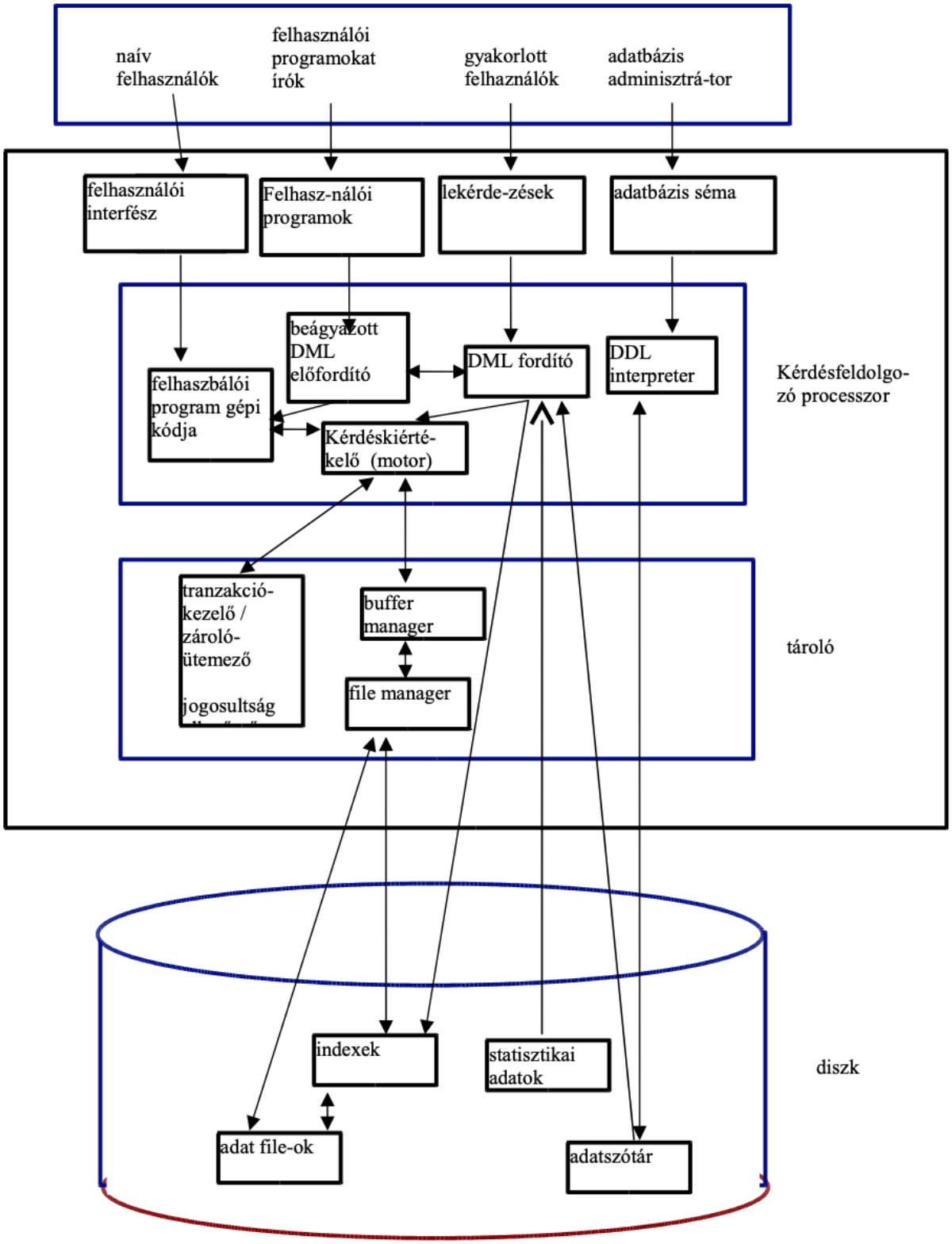
Kliens - szerverarchitektúrák a napjainkban használt kialakítások

Az adatbázis-kezelő rendszer vázlatos felépítése (3. ábra):



3. ábra Adatbázis-kezelő rendszer vázlatos felépítése

Adatbázis-kezelő rendszer egy lehetséges felépítése részletesen [4] (4. ábra)



4. ábra Adatbázis-kezelő rendszer egy lehetséges felépítése részletesen

Összefoglalás

Adatbázis: nagy mennyiségű információ

Adatbázis-kezelő rendszer követelmények:
1. DDL (Data Definiton Language, Adatdefiníciós nyelv)
2. DML (Data Manipulation Language, Adatmanipulációs nyelv)
3. Biztonság
4. Konkurrencia
HATÉKONYSÁG!

Adatbázis-kezelő rendszer története:
- téma szerint:
banki, helyfoglalási, vállalati

- adatmodell szerint:
hálós, hierarchikus, relációs

Adatbázis-kezelő rendszer részei:
- Felhasználói interfész
- Lekérdezés feldolgozó
- Tárkezelő
- Tranzakciókezelő
- Fizikai tároló

Új szavak, fogalmak:
- adatbázis-kezelő rendszer (DBMS)
- DDL, DML
- tranzakció, tranzakciókezelés
- atomosság
- adat, információ
- metaadat
- adatmodell

III. Adatmodellezés

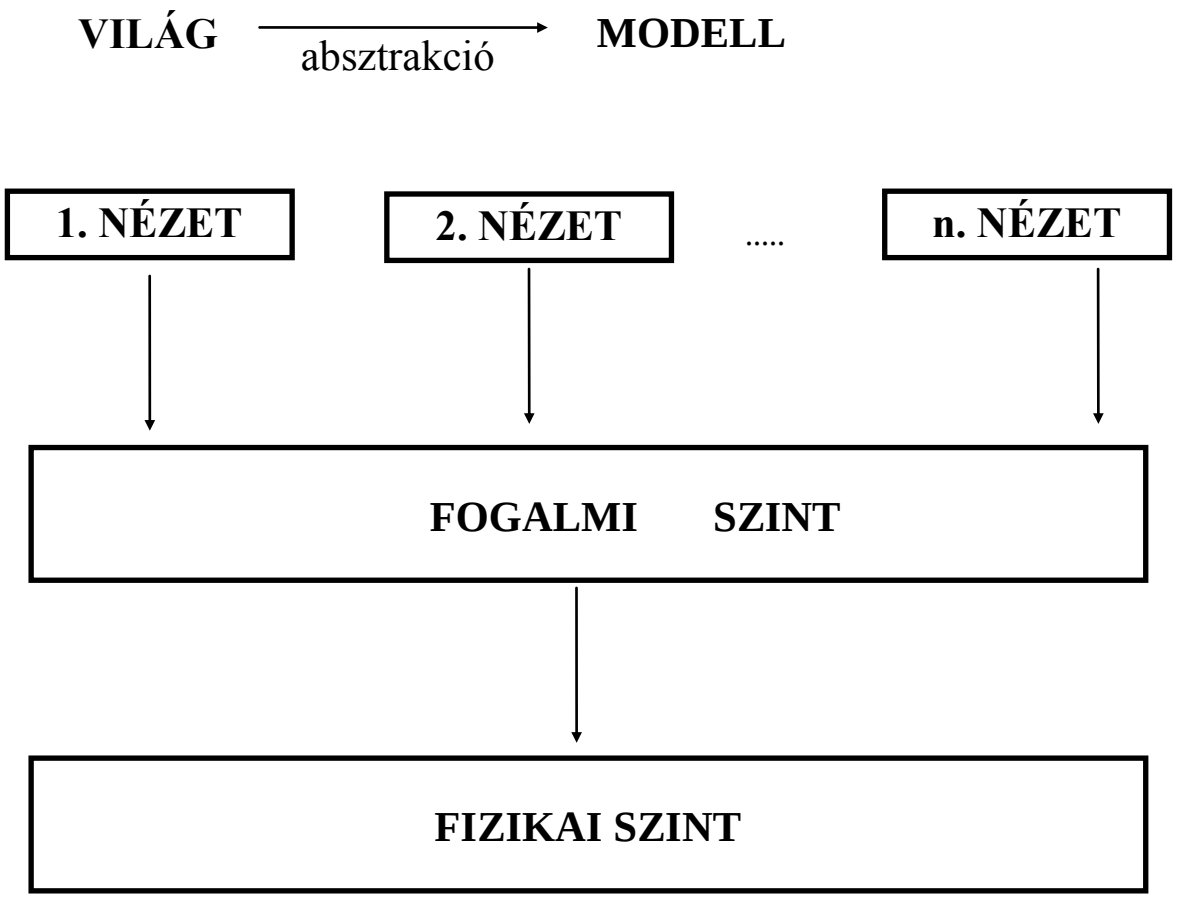
Adatmodellezésnek azt az absztrakciós folyamatot nevezzük, amelyben a valós (mikro)világ tényeit, valamint a tények közötti kapcsolatokat tükröző adatokat, azok lényeges jellemzőit összegyűjtjük, és ezeket számítógépes adaptálásra alkalmas, szabványok által meghatározott formában, az ún. **adatmodellben** rögzítjük.

Az **adatmodell** az adatbázisok szerkezetének leírására szolgál.

Az adatmodellezés nem foglalkozik az adatok konkrét értékeivel, hanem azok belső szerkezetével, és egymás közti kapcsolataival.

Egy adott adatmodellnek sok konkrét adatbázis megfelelhet. Az adatoknak azt a (adatmodellnek megfelelően) strukturált halmazát, amely egy adott pillanatban az adatbázisban tárolva van, az **adatbázis előfordulásának** nevezzük.

A fenti fogalmakat az egyes modellek tárgyalásánál pontosítjuk.



Az adatmodellezés szintje

5. ábra Az adatmodellezés szintjei

A fenti ábrán az adatmodellezés szintjei láthatók (5. ábra)

I. Nézet vagy külső szint:
Az egyes felhasználók "rálátása" az adatbázis bizonyos, hozzáférési jogoktól függő részeire (alsémák - SDDL)

II. Fogalmi vagy logikai szint:
Az egész adatbázisra vonatkozó **adatmodell** (séma - DDL),

III. Fizikai vagy belső szint:
Tárolás és elérés módjai: fájlok, indexek egyéb tárolási szerkezetek.

Adatfüggetlenség:

logikai:
logikai szinten az adatbázis leírása megváltoztatható anélkül, hogy a fizikai szinten változás történne

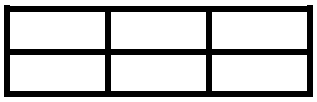
fizikai:
fizikai szinten az adatbázis leírása megváltoztatható anélkül, hogy a logikai szinten változás történne

Példa

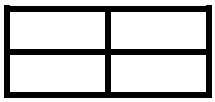
Tekintsünk egy m x n-es tömböt. Az adatmodellezés három szintjét ezen a példán következőképpen illusztrálhatjuk:

Belső (vagy fizikai) szint: m x n memóriahely

Logikai (vagy fogalmi) szint:



Külső (vagy nézet) szint: 2^(m x n) lehetséges nézet



Fogalmi szinten a megfelelő **séma** pl.:

```
type tgrid:=array[1..2, 1..3] of integer;
var vgrid1, vgrid2:=tgrid;
```

Egy **előfordulás**:

1	2	3
4	5	6

Fogalmi/Logikai szint

Objektum fogalmon alapú modellek:

- **Egyed-kapcsolat modell** (grafikus szabvány)

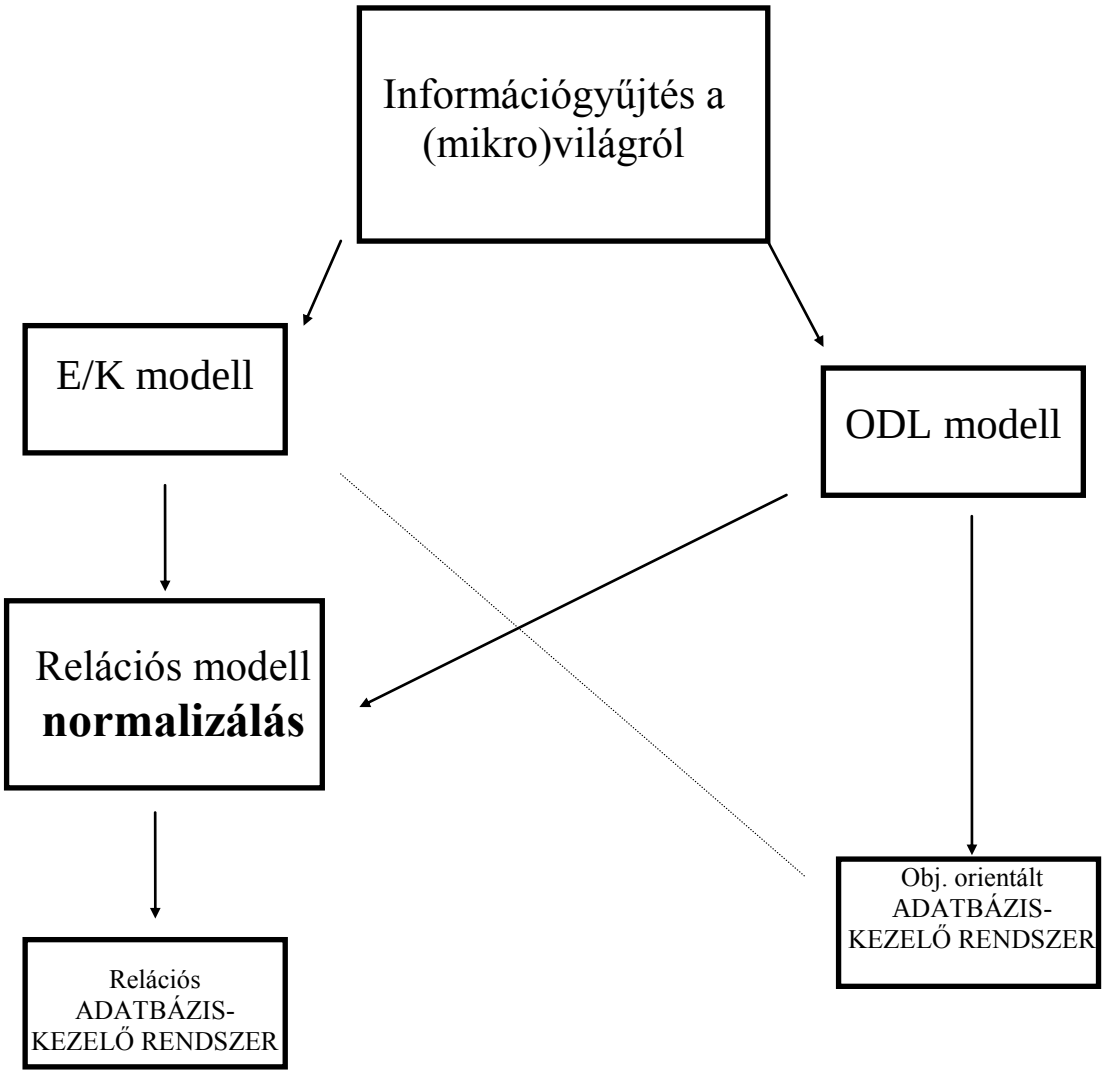
- Objektumorientált modell, ODL (C++ nyelvhez hasonló szabvány az adatstruktúrák leírására)

Rekord fogalmon alapuló modellek:

- **Relációs modell**
- Hálós modell
- Hierarchikus modell

Ebben a könyvben az Egyed-Kapcsolat modellel és a relációs modellel foglalkozunk [5].

Az adatbázis-modellezés és implementálás folyamata a gyakorlatban (6. ábra)



6. ábra Az adatbázis-modellezés és implementálás folyamata a gyakorlatban

IV. Egyed - kapcsolat diagramok

Egyedhalmazok: elemei az **egyed**ek. (osztály)
Egyed: a világban létező dolog (objektum: személy, tárgy, fogalom, stb.)
Jele: téglalap

Tulajdonság (vagy attribútum): az egyed jellemző jegye. A tulajdonságok adnak lehetőséget az egyes egyedek megkülönböztetésére.
Jele: ellipszis, vonallal kapcsolódik az egyedhez

Kapcsolat: az egyedek logikai összefüggése, két vagy több egyedhalmazt kapcsol össze.
Jele: rombusz, vonallal vagy nyíllal kapcsolódik a résztvevő egyedekhez. Nyilat az N: 1 kapcsolat "1" oldalán használunk.

Kapcsolatok típusai:

a.) Egy - egy (1:1): vállalkozók - adószámok
b.) Egy - sok (1:N): édesanyák - gyerekek (az "1" -re mutató **nyíl** használata)
c.) Sok - sok (M:N): filmek - színészek

A kapcsolat reprezentálása lehetséges pl. **reláció**kkal. A kapcsolathalmaz elemei az összetartozó (relációban levő) párok. A pontos definíciót később közöljük.

a.)

Kisnagy Valér	12345678
Mándoki Itala	01234567
Nagykis Róza	01010101

b.)

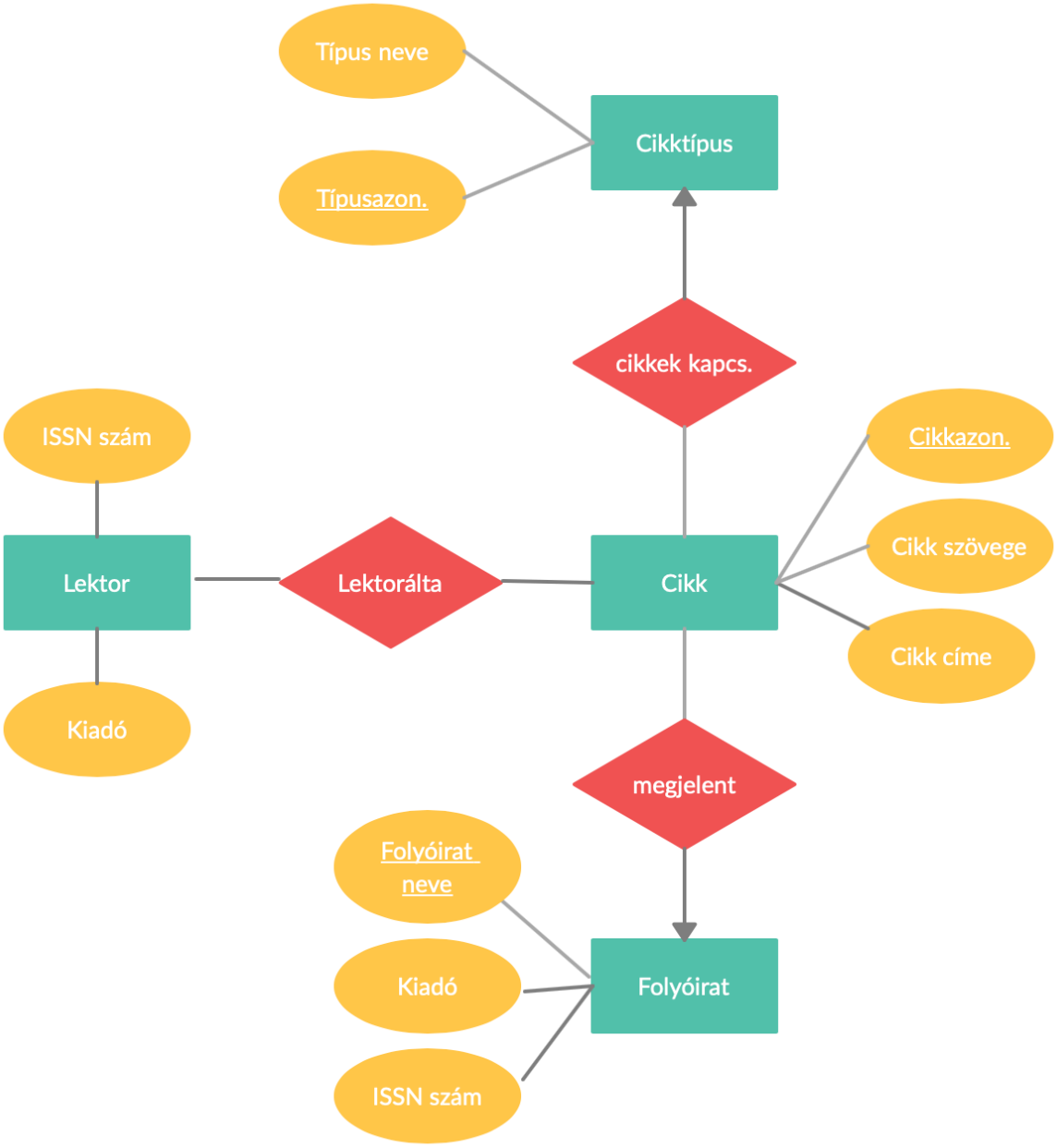
Kisnagyné B. Valéria	Kisnagy László
Mándoki Itala	Mándoki Alvin
Kisnagyné B. Valéria	Kisnagy Dániel
Kisnagyné B. Valéria	Kisnagy Kata

c.)

Elemi ösztön	Sharon Stone
Emlékmás	Arnold Schwarzenegger
Elmlékmás	Sharon Stone

Példa:

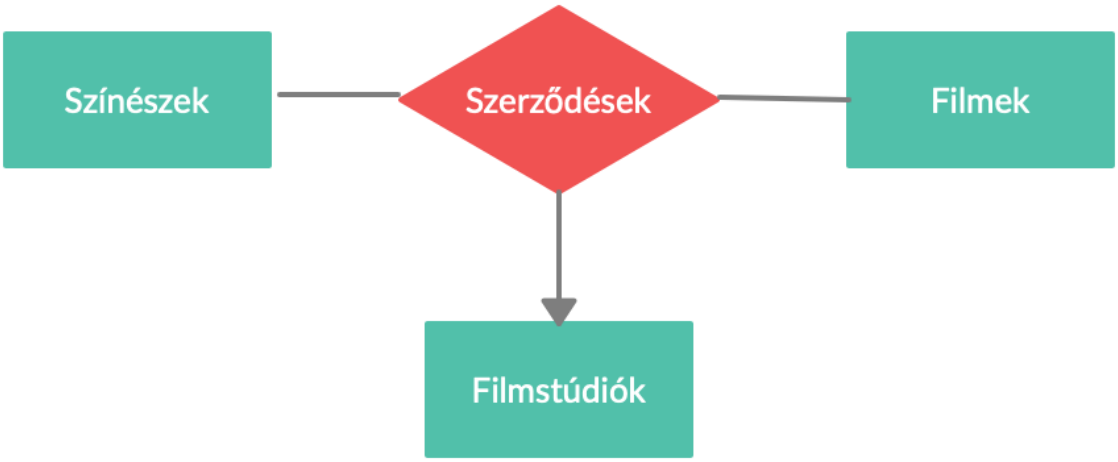
A 7. ábrán egy összetett Egyed-Kapcsolat modellt láthatunk



7. ábra összetett Egyed-Kapcsolat modell

Példa:

Sokágú Egyed-Kapcsolatra mutat példát a 8. ábra



8. ábra Sokágú Egyed-Kapcsolat modell

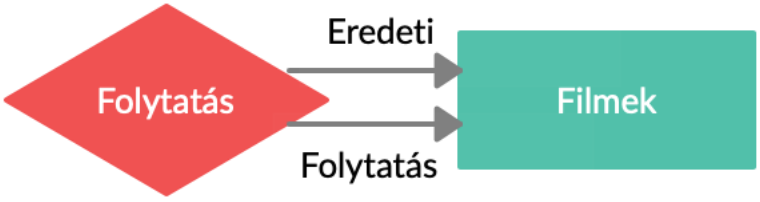
A stúdiót nyilván a film maga egyértelműen meghatározza, de ennek jelölésére az E/K modellben nincsen mód (8. ábra).

A Szerződések kapcsolat kapcsolathalmaza a (Színész, Stúdió, Film) hármasokból áll.

Szerepek a kapcsolatokban

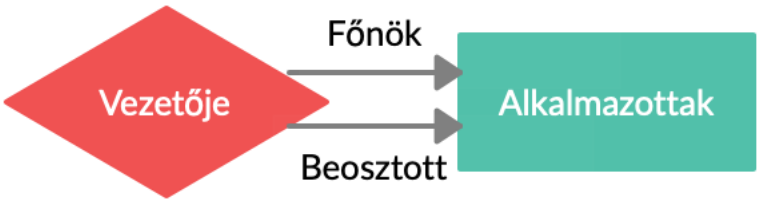
Egy egyedhalmaz kétszer, vagy többször is részt vehet egy kapcsolatban.
Ilyenkor több éllel kötjük a kapcsolathoz, és az élre ráírjuk a szerep nevét (9. ábra), (10 ábra).

Példa:



9. ábra Többélű kapcsolat példa

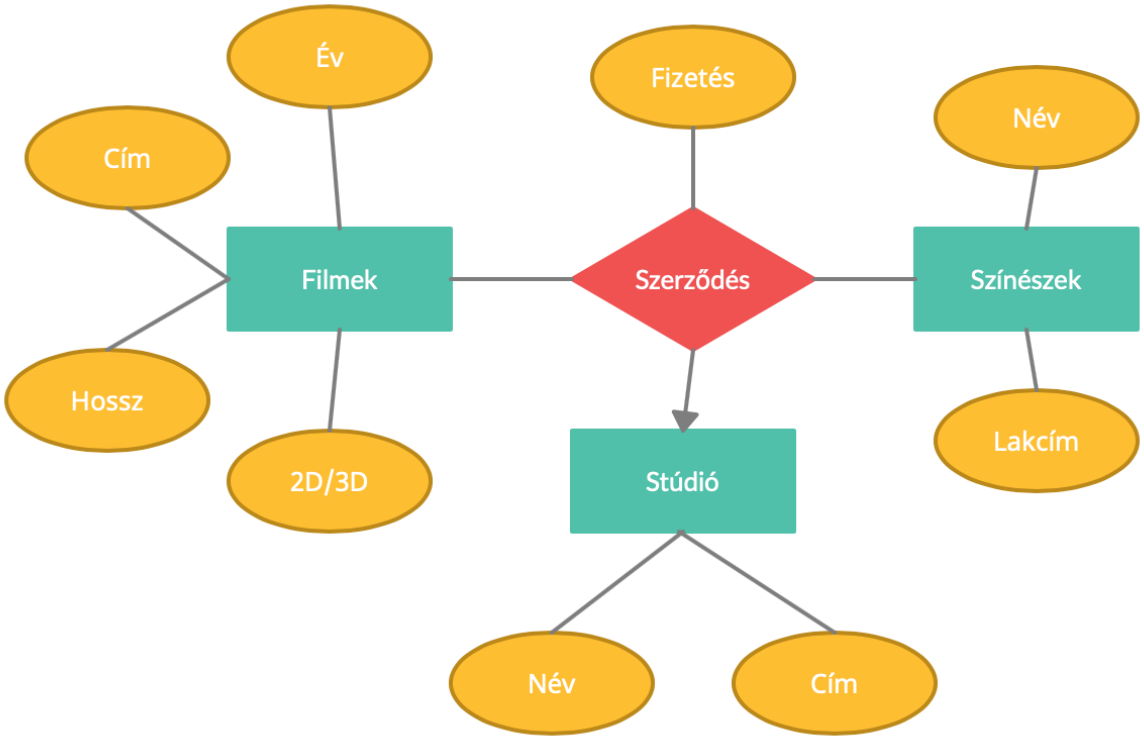
Példa:



10. ábra Többélű kapcsolat példa2

Példa:

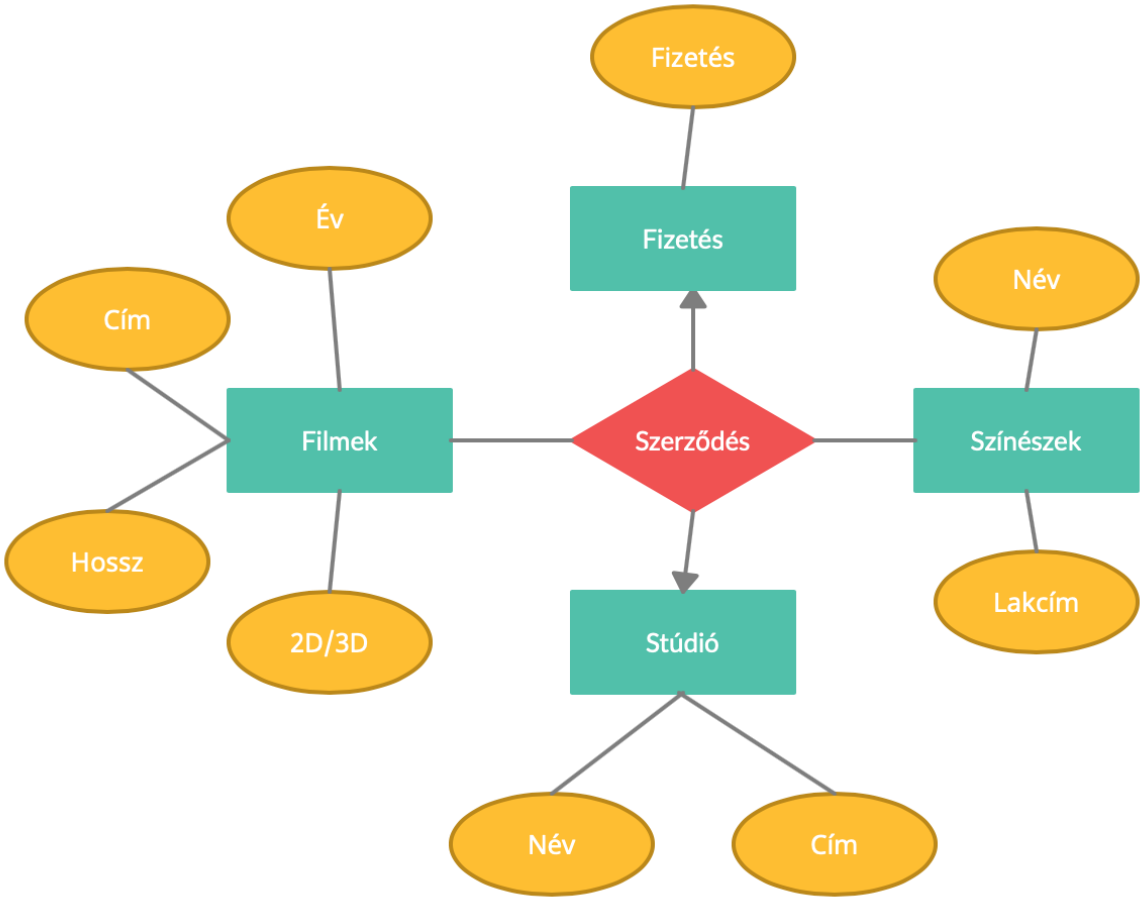
A kapcsolatok attribútumait mutatja be a 11. ábra.



***11**. ábra Kapcsolatok attribútumai*

Példa:

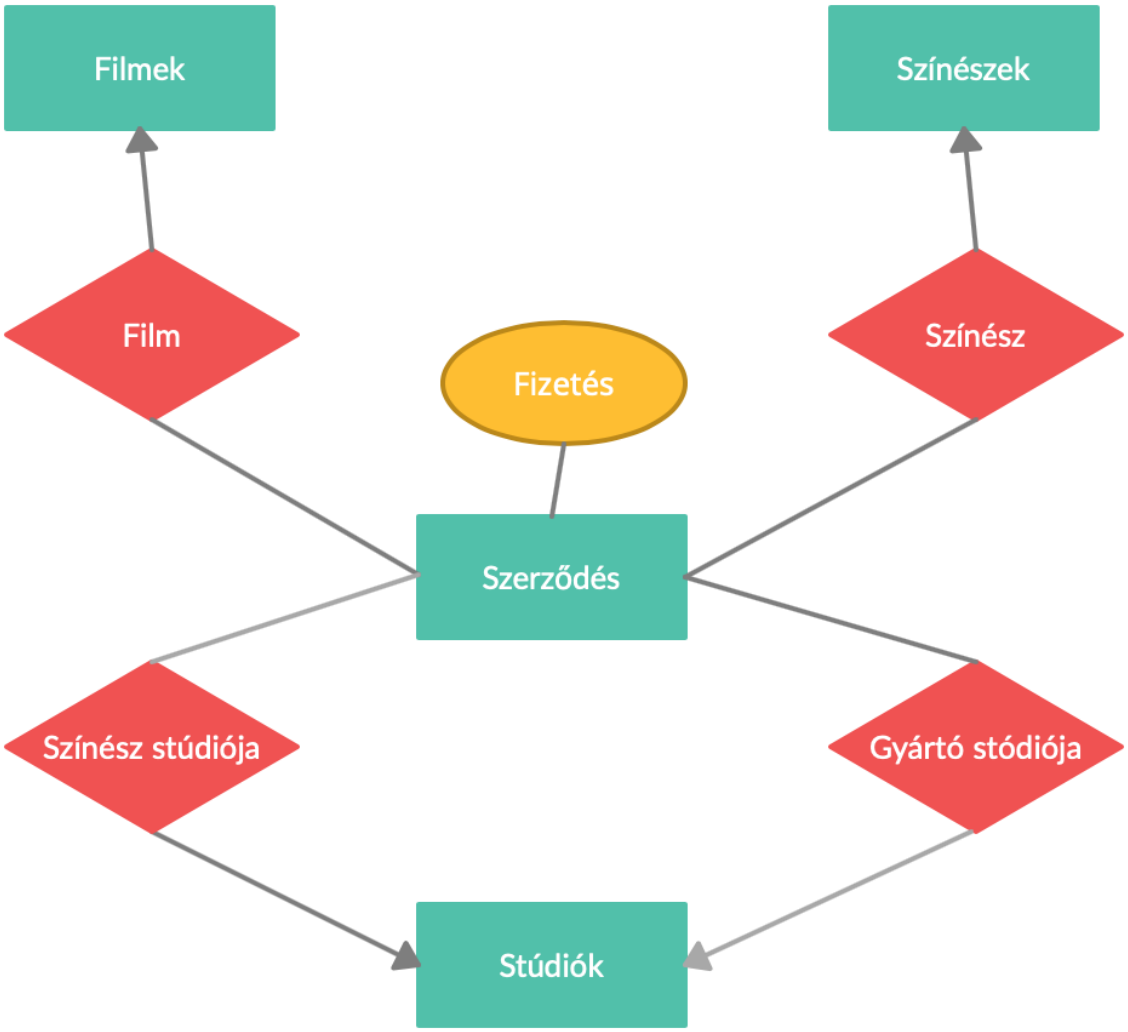
Kapcsolat attribútumának megszüntetése egyedhalmaz közbeiktatásával látható a 12. ábrán



***12**. ábra Kapcsolat attribútumának megszüntetése egyedhalmaz közbeiktatásával*

Példa:

Többágú kapcsolat átalakítása bináris kapcsolatokká kapcsoló egyedhalmaz közbeiktatásával látható a 13. ábrán



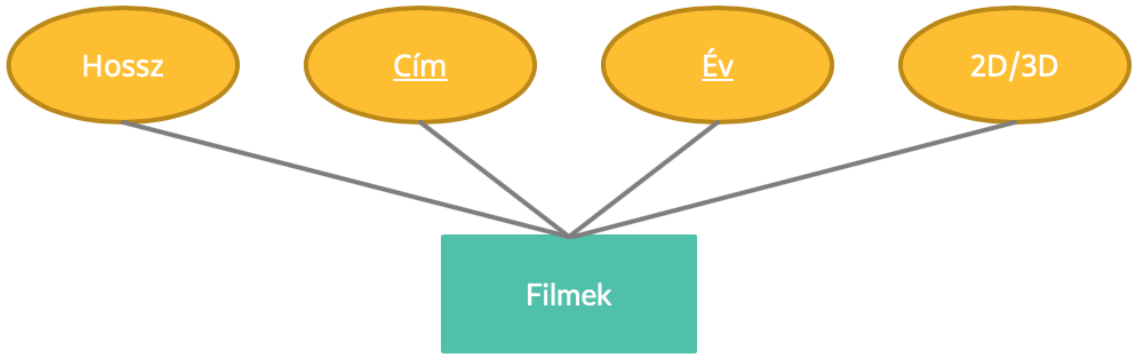
***13**. ábra Többágú kapcsolat átalakítása bináris kapcsolatokká kapcsoló egyedhalmaz közbeiktatásával*

Megszorítások modellezése

A megszorítások, olyan szabályok, melyek az adatmodell létrehozása során kötelezően figyelembe veendők.

Megszorítások:
1.
Szuperkulcs: olyan attribútumhalmaz, amely az egyedet egyértelműen meghatározza
Kulcsjelölt: olyan szuperkulcs, amelynek egyetlen valódi részhalmaza sem kulcs.
Elsődleges kulcs: több kulcsjelölt esetén az, amelyiket ténylegen használni fogunk. Szokás röviden **kulcs**-nak nevezni.

Az E/K ábrán az elsődleges kulcsot jelöljük, az attribútum aláhúzásával (14. ábra). Egy filmet a címe egyedül nem feltétlenül tud meghatározni, szükséges az év attribútum értéke is az egyértelműsítéshez.

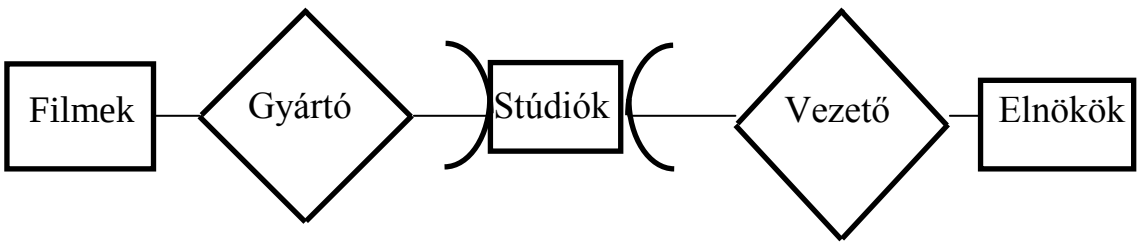


14. ábra E/K ábrán az elsődleges kulcsot jelöljük, az attribútum aláhúzásával

2.
Egyértékűség:
az egyik egyedhalmaz egyes egyedeihez a másik egyedhalmazból pontosan egy kapcsolódik. Ezt az N:1 kapcsolatban az 1 oldalra mutató nyíllal jelöljük.

3.
Hivatkozási épség: amire hivatkozunk, ott nem állhat NULL érték (not null)
NULL érték: az adott attribútum értéke ismeretlen
Jelölés: gömbölyű nyíl

Példa: minden filmhez egy stúdió, minden elnökhöz egy stúdió tartozik (15. ábra).



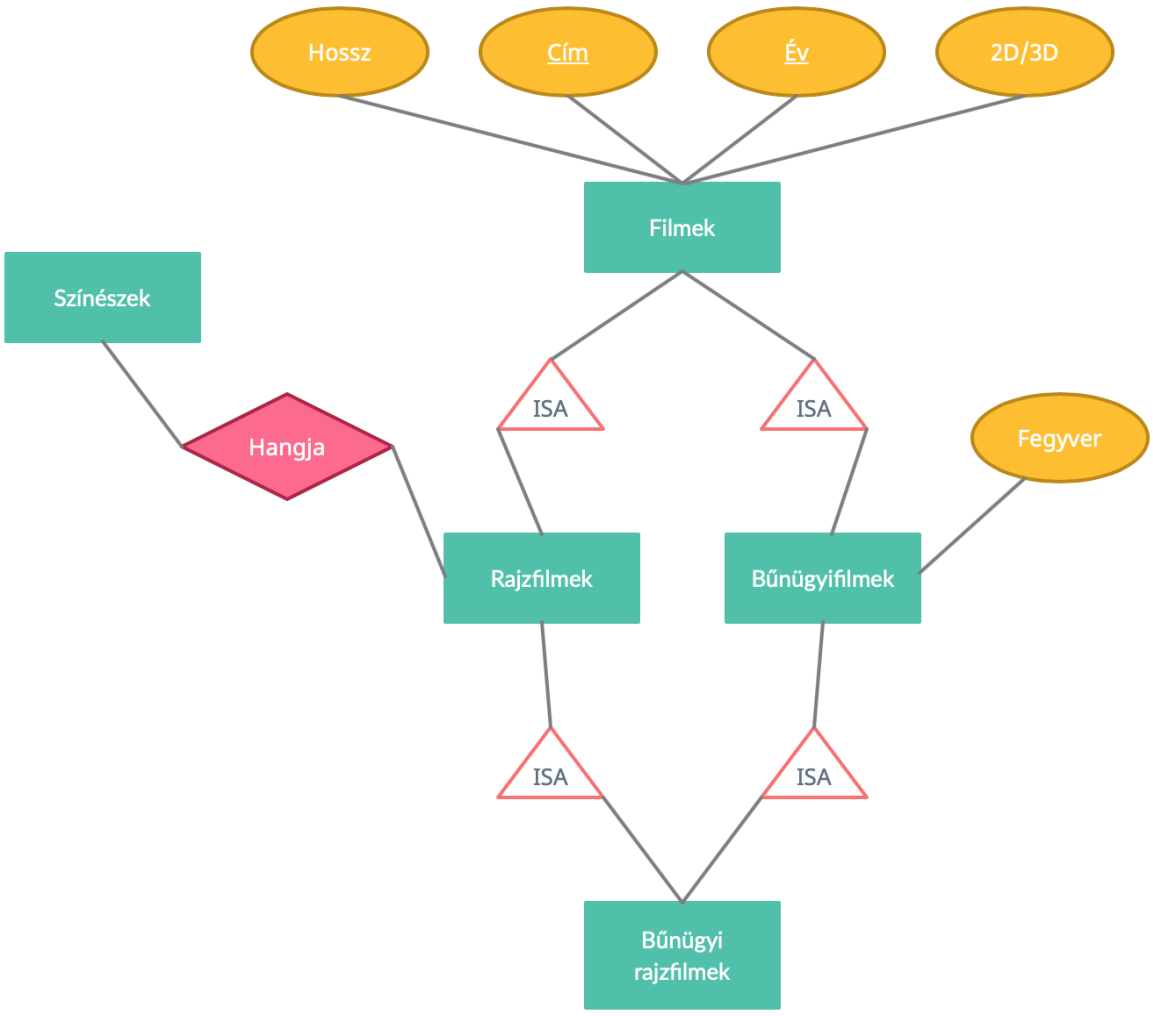
***15**. ábra Minden filmhez egy stúdió, minden elnökhöz egy stúdió tartozik*

4.
Értelmezési tartományra vonatkozó megszorítások:
Pl. mely értékhatárok közé eshet az adott attribútum.

5.
Egyéb megszorítások: Pl. egy kapcsolatban az összetartozó egyedek számára vonatkozó korlátok, melyet az egyedhalmazokat összekötő vonalak címkézésével jelölhetünk.

Alosztályok, öröklődés az E/K modellben

Az alosztályok esetében a tábla összes attribútuma öröklődik a származtatott táblába. A származtatást háromszöggel jelöljük és „az egy" felirattal. A származtatott táblánál már csak azokat az attribútumokat jelöljük, melyek új attribútumként jelennek meg a táblában (16. ábra).



***16**. ábra Alosztályok, öröklődés az E/K modellben*

Gyenge egyedhalmazok

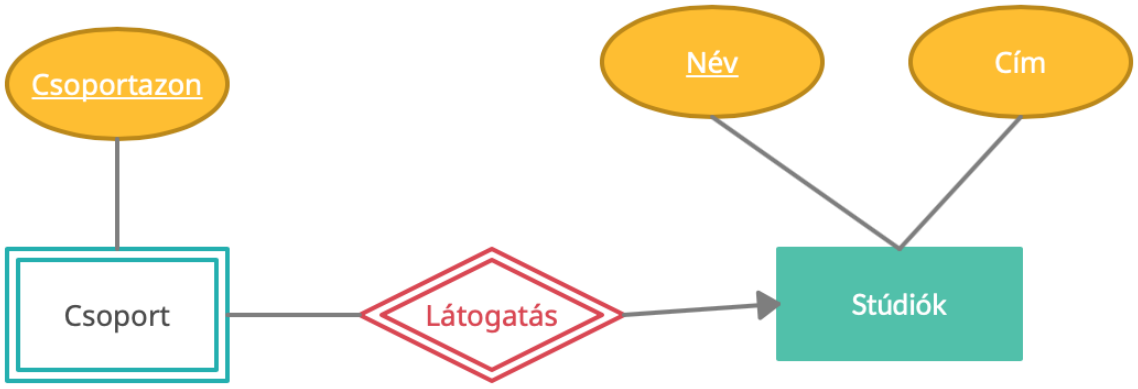
Gyenge egyedhalmaz: a hozzátartozó attribútumok nem elégségesek az egyedek azonosításához. Ezért a kulcsban szerepelnek más egyedhalmazok attribútumai is.

Jelölés: dupla vonallal keretezzük mind az egyedhalmazt, mind a kapcsolatokat, amiben részt vesz.

Okai:

1. Az egyedhalmaz része egy másiknak (17. ábra)

Példa:

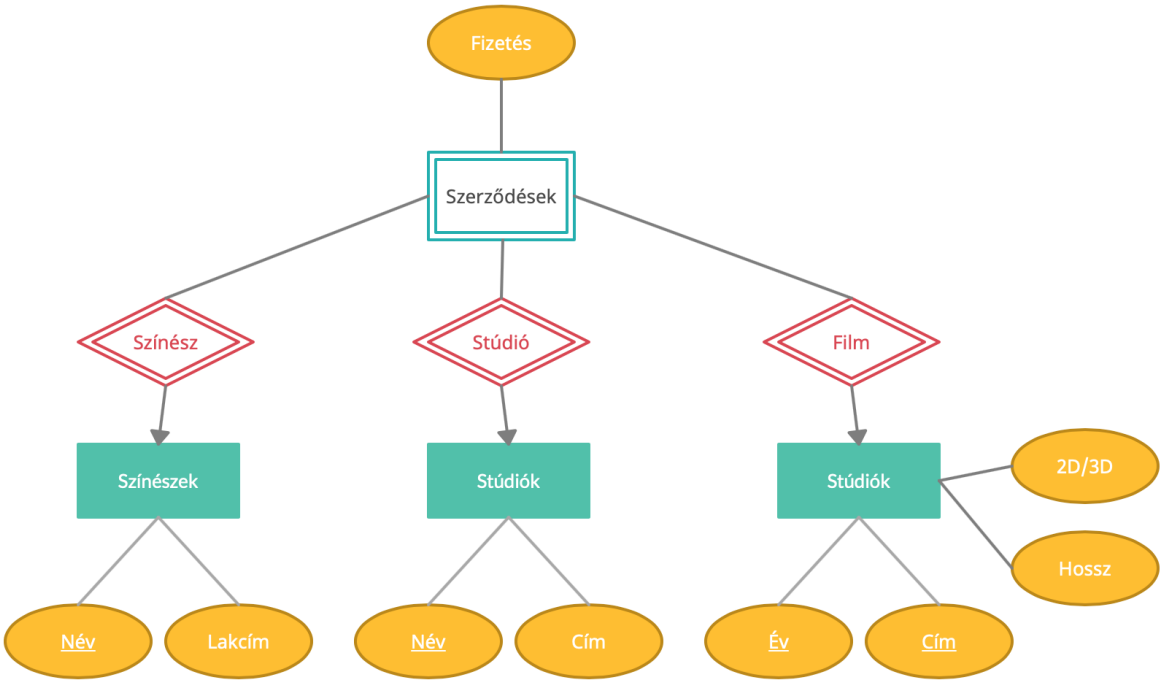


17. ábra Gyenge egyedhalmaz

2. Sokágú kapcsolatok binárissá alakítása kapcsoló egyedhalmaz segítségével.

Ekkor a kapcsoló egyedhalmaz legtöbbször gyenge (18. ábra).

Példa:



18. ábra Sokágú kapcsolatok binárissá alakítása

Gyenge egyedhalmazokra vonatkozó követelmények

1. A kapcsolatnak N:1 típusúnak kell lennie, melynek az 1 oldala a nem gyenge egyedhalmaz.

2. A kapcsolódó egyedhalmaz kulcsának tartalmaznia kell a gyenge egyedhalmaz kulcsát.

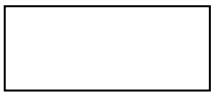
3. Ha a kapcsolódó egyedhalmaz szintén gyenge, akkor arra is teljesülnie kell 1.-nek és 2.-nek.

4. Ha több sok-egy kapcsolat is vezet a gyenge egyedhalmazból más (nem gyenge) egyedhalmazokhoz, akkor mindegyik kapcsolaton át segíthető a gyenge egyedhalmaz kulcsának kialakítása.

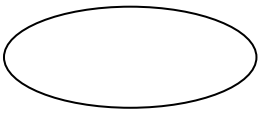
Összefoglalás

Az Egyed-Kapcsolat diagramm jelölései

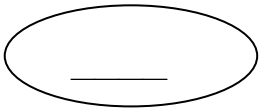
A korábbi ábrákon a színeknek nincs jelentősége csak a könnyebb megkülönböztetés miatt használtuk.



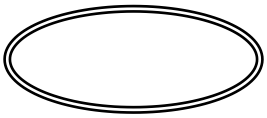
Egyedhalmaz



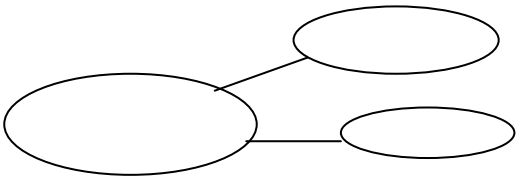
Attribútum



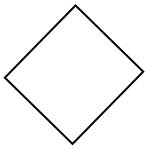
Kulcsattribútum



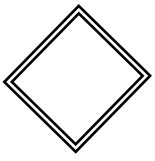
Többértékű attribútum (halmaz, lista)



Összetett attribútum (reláció)

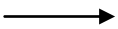


Kapcsolathalmaz



Gyenge kapcsolathalmaz

N:1 kapcsolat (Nyíl 1-re mutat)



Tervezési alapelvek

Amikor egy feladathoz tartozó adatbázist tervezünk meg, az alábbi alapelveket kell betartanunk.

1. Valósághű modellezés

2. Redundancia elkerülése

3. Egyszerűség

4. Megfelelő elem választása (attribútum vagy egyedhalmaz?)

A relációs modell

A reláció fogalma

Matematikai definíció:

Halmazok Descartes szorzata (direkt szorzata):

Legyenek D_1, D_2, …D_n adott halmazok. E halmazok

Descartes szorzata: $D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots d_n) \mid d_k$ eleme az D_k halmaznak, $1 \leq k \leq n \}$

A direkt szorzat tehát olyan rendezett érték n-eseket tartalmaz, amelynek k. eleme a k. halmazból való.

Példa:

Filmek:= {Elemi ösztön, Emlékmás, 6:3}
Színészek:= {Sharon Stone, Arnold Schwarzenegger, Eperjes Károly}

Filmek x Színészek = {(Elemi ösztön, Sharon Stone),
(Elemi ösztön, Arnold Schwarzenegger),
(Elemi ösztön, Eperjes Károly),
(Emlékmás, Sharon Stone),
(Emlékmás, Arnold Schwarzenegger),
(Emlékmás, Eperjes Károly),
(6:3, Sharon Stone),
(6:3, Arnold Schwarzenegger),
(6:3, Eperjes Károly)}

Reláció:

Relációnak nevezzük az $D_1 \times D_2 \times \ldots \times D_n$ direkt szorzat bármely részhalmazát:

$r \subseteq D_1 \times D_2 \times \ldots \times D_n$

Példa: { (Elemi ösztön, Sharon Stone),
(Emlékmás, Sharon Stone),
(Emlékmás, Arnold Schwarzenegger),
(6:3, Eperjes Károly) }

Az D_1, D_2, …D_n adott halmazok a reláció ún. **(érték)tartományai** (angolul Domain).

A relációt szokás táblázatban ábrázolni. A reláció egy eleme a táblázat egy **sora** .

A halmaz fogalmából következik, hogy

- a reláció olyan táblázatnak tekinthető, melynek nem lehetnek azonos sorai. (A sort szokás rekordnak is nevezni).
- a sorok sorrendje tetszőleges.

A gyakorlatban a relációkat e szokásos módon, táblázattal jelöljük. Az adatbázis kontexusban egy konkrét táblázatot a **reláció egy előfordulásának**, vagy **példányának** nevezünk.

A matematikai definícióból nem következik, hogy a relációt reprezentáló táblázathoz fejlécet kellene készíteni. A relációs adatbázisok kezelésének azonban alapvető feltétele, hogy az értéktartományoknak megfelelő absztrakt jellemzők neveit felhasználjuk a sor elemeinek jelölésére. Az értéktartományok elemeit felvevő jellemzők az ún. **attribútumok**. A táblázatos ábrázolásban az attribútumokat a reláció első sorába írhatjuk.
Ez az első sor lesz a **táblázat fejléce**. Következésképpen a táblázat sorainak a fejléctől különböző sorok tekinthetők.

Példa:

FILM	SZINÉSZ
Elemi ösztön	Sharon Stone
Emlékmás	Sharon Stone
Emlékmás	Arnold Schwarzenegger
6:3	Eperjes Károly

A relációs modellben a relációt a **séma**val jellemezzük.

Relációséma: A reláció neve, és az attribútumok halmaza együttesen alkotja a reláció sémáját.
Jelölés: Relációnév(attribútum1, attribútum2,…)

Példa: Filmszínészek(filmcím, színész neve)

Relációs adatbázisséma: A relációsémákból álló halmaz.

Az adatbázis fogalmi leírását az adatbázisséma és az attribútumok kapcsolatatainak (függőségeinek) halmaza együttesen adja.

A kapcsolatok leírásával egy későbbi előadásban foglalkozunk, azonban a legfontosabb kapcsolatot itt is megemlítjük, és ez a **reláció elsődleges kulcsa**. Ugyanúgy, mint az E/K modellben, itt is a szuperkulcs, kulcsjelölt, elsődleges kulcs fogalmakkal dolgozunk.

Elsődleges kulcs jelölése: az attribútum nevének aláhúzása

Példa:

Legyen a reláció neve "Adatok".
Sémája: Adatok (AZON, NÉV, IRSZ, VÁROS, SZÜL_DAT)

Az aláhúzott attribútum a reláció elsődleges kulcsa.
Az egyszerűség kedvéért tegyük fel, hogy a lehetséges értéktartományok mindegyike a legfeljebb 30 karakter hosszúságú sztringek halmaza.

A reláció egy előfordulása: hatosok halmaza, pl.:

<0524, Kovács Zoltán, 4028, Debrecen, Kút u.32., 12-AUG-46>
<0525, Tar Ede, 4090, Polgár, Kerek u. 96., 22-JAN-70>
<0526, Villám Éva, 4029, Debrecen, Kasssai u. 55, 122-JAN-70>

A reláció példány táblázatban is ábrázolható:

0524	Kovács Zoltán	4028	Debrecen	Kút u.32.	12-AUG-46
0525	Tar Ede	4090	Polgár	Kerek u.96.	03-JAN-40
0526	Villám Éva	4029	Debrecen	Kassai u.55.	22-JAN-70

A reláció sémája a táblázat fejlécébe írható:

AZON	NEV	IRSZ	VAROS	UTCA	SZUL_DAT
0524	Kovács Zoltán	4028	Debrecen	Kút u.32.	12-AUG-46
0525	Tar Ede	4090	Polgár	Kerek u.96.	03-JAN-40
0526	Villám Éva	4029	Debrecen	Kassai u.55.	22-JAN-70

A reláció sorának formális fogalma: A sor az attribútumokon értelmezett függvény, melynek értékkészlete az attribútumoknak megfelelő értéktartományok direkt szorzatának egy eleme.

Következmény: az attribútumok sorrendje tetszőleges

Példa:

Egy lehetséges függvény: Film → Elemi ösztön
Színész → Sharon Stone

Egy másik lehetséges függvény: Film → 6:3
Színész → Eperjes Károly

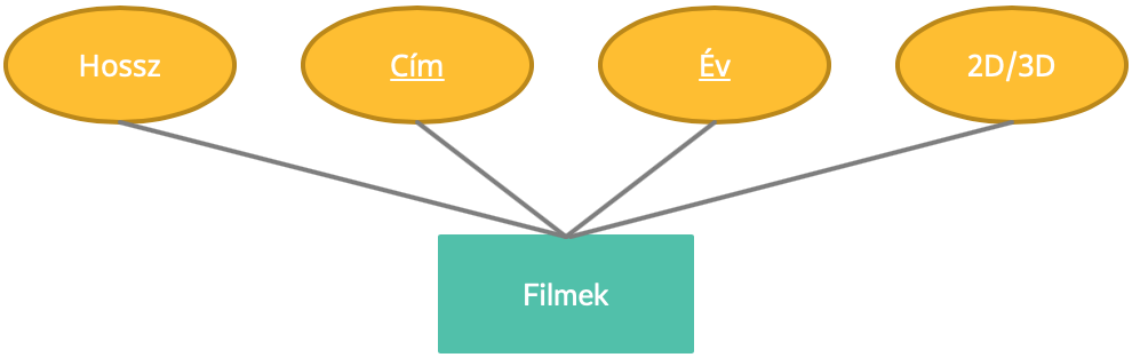
V. E/K diagram átírása relációkká

Egyedhalmazok átírása relációkká:

A reláció kulcsa az egyedhalmaz kulcsa.

Minden (**nem gyenge) egyedhalmaz**nak megfeleltetünk egy relációt, melynek neve az egyedhalmaz neve, attribútumai pedig az egyedhalmaz attribútumai (19. ábra). A reláció kulcsa az egyedhalmaz kulcsa.

Példa:



19. ábra E/K diagram átírása relációkká

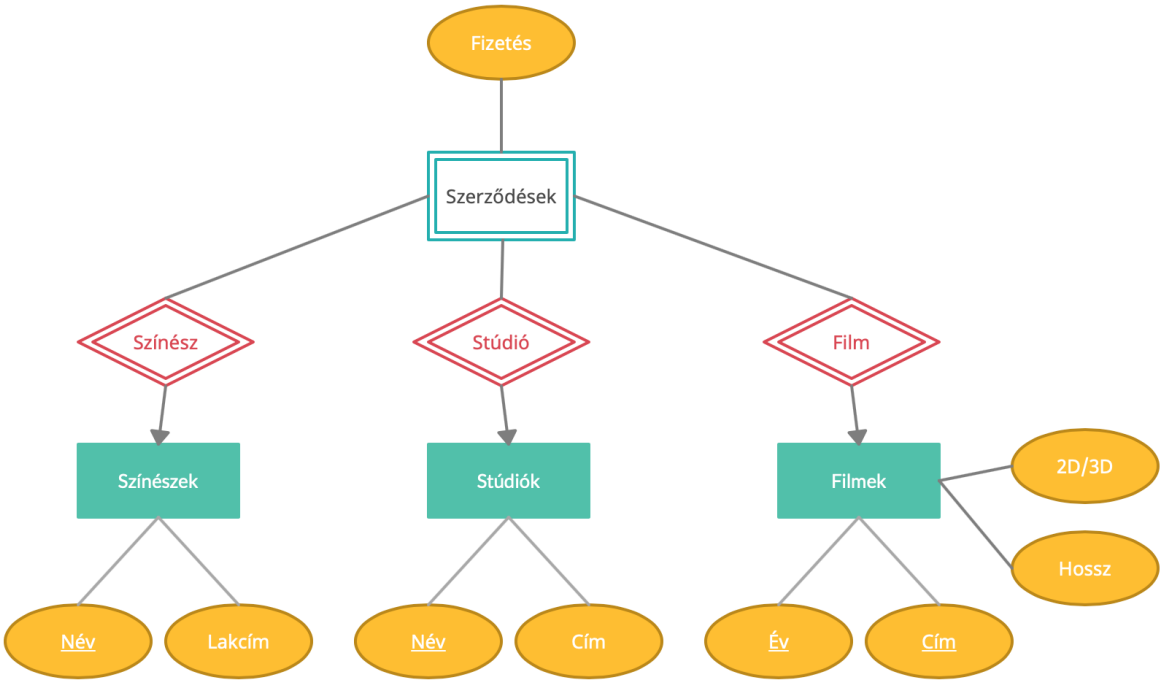
Az eredmény reláció:

Filmek(Hossz, Cím, Év, 2D/3D)

A reláció nevét leírjuk és mögé zárójelben felsoroljuk a hozzátartozó attribútumokat. A kulcs mezőket itt is aláhúzással jelöljük.

Egyedhalmazok átírása: Minden **gyenge egyedhalmaz**nak megfeleltetünk egy relációt, melynek attribútumai a gyenge egyedhalmaz attribútumai és a hozzá N:1 kapcsolatban kapcsolódó egyedhalmaz(ok) azon attribútuma(i), mely(ek) a gyenge egyedhalmaz kulcsában részt vesz(nek).

Példa: Gyakran az attribútumokat át kell nevezni (20. ábra):



20. ábra Gyenge egyedhalmaz áítása relációkká

Az eredmény reláció:

Szerződések(SzínészNév, StúdióNév, FilmCím, Év, Fizetés)

Kapcsolatok átírása relációkká:

Minden **erős egyedhalmazok közti kapcsolathoz** rendelünk egy relációt.
A reláció attribútumai a kapcsolatban résztvevő egyedhalmazok kulcsattribútumai.

Kulcsok:

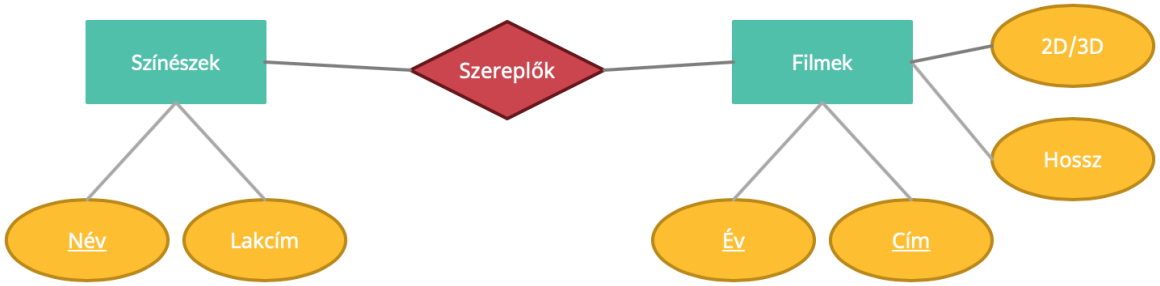
1: n kapcsolat: n oldalon álló egyedhalmaz kulcsa
(1:1 esetben tetszőleges)

m:n kapcsolat: a résztvevő egyedosztályok kulcsainak egyesítése.

Mivel a gyenge egyedhalmazhoz rendelt relációban a hozzákapcsolódó erős egyedhalmazok kulcsa is szerepel a relációsémában, ezért a **gyenge kapcsolatnak** nem feleltetünk meg relációt. (Ld. előző példa)

Kapcsolatok átírása (21. ábra):

Példa:



21. ábra Kapcsolatok átírása relációkká

Az eredmény adatbázis séma:

Filmek(FilmCím, Év, Hossz, 2D/3D)
Színészek(SzínészNév, Lakcím)
Szereplők(FilmCím, Év, SzínészNév)

Minden egyedhalmaznak és minden ERŐS kapcsolatnak megfelel egy reláció!

Kapcsolatok átírása (folytatás):

A **specializáló kapcsolat** reprezentálására két alapvető módszer van:

1. Az eddigieknek megfelelően minden egyedhalmazhoz rendelünk egy-egy relációt, de az egyes egyedhalmazok attribútumai közé felvesszük a szuperosztály kulcsát is. Igy e közös attribútum fejezi ki a specializálást, és egyben ez teszi lehetővé az adott egyed minden tulajdonságának elérését. A kulcs a szuperosztály kulcsa.

Példa:

A Filmek szuperosztály egyedhalmazának relációsémája:

Filmek(Cím, Év, Hossz, Szalagfajta)

A Bűnügyi filmek alosztály egyedhalmazának relációsémája:

Bűnügyi filmek(Cím, Év, Fegyver)

2. Az alosztályok jellemzésére bevezethető egy-egy új attribútum a meglevő szuperosztály attribútumain kívül. Ezen attribútumok értéke lehet kitöltött vagy NULL érték aszerint, hogy a szóbanforgó egyed tagja-e az attribútummal jellemzett alosztálynak.

Problémák:

- A NULL érték szerepe (osztályozó, vagy ismeretlen érték)
- Attribútumok magas száma

E/K modellből átírt relációk kulcsainak képzésére vonatkozó szabályok összefoglalása

1. Ha a reláció egyedhalmaz átírásából keletkezett, akkor a kulcs az egyedhalmaz kulcsattribútumainak halmaza.

2. A kapcsolatok átírásából keletkező relációk esetében a kapcsolatok típusai szerint a következőképpen kapjuk a reláció kulcsát (feltesszük, hogy csak bináris kapcsolatokról van szó):

- N:M kapcsolatnál mindkét résztvevő egyedhalmaz kulcsának egyesítése
- N:1 kapcsolatnál az N oldalon álló egyedhalmaz kulcsa
- (1:1 kapcsolatnál egyik (választott) egyedhalmaz kulcsa)

VI. Az adatbázis sémájának normalizálása

Az E/K diagramok átírásából kapott relációsémák még nem véglegesek. Az adatbázis tervezésének következő szakasza az egyes relációsémákban szereplő attribútumok egymás közötti kapcsolatainak ellenőrzése.
A kapcsolatokat ún. függőségekkel írjuk le. Az ellenőrzés, és ha szükséges, az azt követő sémaátalakítás folyamatát **normalizálás**nak nevezzük.
A normalizálás célja, hogy a lehető legkisebbre csökkentsük az adatbáziskezelő rendszerben a használat során előforduló potenciális hibaforrásokat. A normalizálással a későbbiekben foglalkozunk.

Funkcionális függőségek

Kulcsfüggőség általánosítása

Definíció:
Az R (A_1, A_2 ,…, A_n) sémájú reláció B $\in$ {A_1, A_2, ...A_n} attribútuma **funkcionálisan függ** az A_1, A_2, ...A_k attribútumoktól, ha mindazon esetekben, amikor R valamely sorai megegyeznek az A_1, A_2, ...A_n attribútumokon, akkor ezek a sorok megegyeznek a B attribútumon is.

Jelölés: A_1, A_2, ...$A_n \rightarrow B$

A sor formális definíciója alapján jelölje t[A] a t sornak az A attribútumon felvett értékét.
Például t[név]='Ullman'.

Definíció:
Ha $t_1[A_1, A_2, ...A_k] = t_2[A_1, A_2, ...A_k]$ teljesülése maga után vonja $t_1[B] = t_2[B]$-t, akkor A_1, A_2, ...A_k funkcionálisan meghatározza B-t.

Könnyen elképzelhetjük a funkcionális függést, ha belegondolunk a következő példába.
Amikor vásárolunk egy boltban, a pénztárnál a termék vonalkódját olvassák be és a kijelzőn, valamint a blokkon rögtön megjelenik a termék neve és egységára, azaz az alábbi funcionális függés írható le:

vonalkód $\rightarrow$ terméknév, egységár

Tehát a vonalkód értéke, meghatározza egyértelműen a termék nevét és egységárát.

Példák

1.
A Nyilvántartás (TAJ-szám, név, cím, fizetés) relációban a

TAJ-szám $\rightarrow$ név, cím, fizetés

funkcionális függőség teljesülését célszerű előírni. Semmelyik más attribútumhalmaz nem határozza meg a tőle különböző attribútumokat.

2.
A Film2(cím, év, hossz, szalagfajta, stúdióNév, stúdióCím)
relációban a következő függőségek írják le az adatok közötti kapcsolatokat:

cím, év →hossz, szalagfajta, stúdióNév, stúdióCím
stúdióNév→ stúdióCím

3. Film(cím, év, hossz, szalagfajta, stúdióNév, színészNév)
relációban a következő függőség érvényes:

színészNév, cím, év →hossz, szalagfajta, stúdióNév

A funkcionális függőség tulajdonságai: ARMSTRONG AXIÓMÁK

Jelölés: α, β, γ a reláció attribútumaiból képezett tetszőleges halmazok.

1. **Reflexív**: Bármely $\beta \subseteq \alpha$ esetén $\alpha \rightarrow \beta$

Definíció: Az A_1, A_2, ...$A_n \rightarrow B_1, B_2, ...B_k$ függőséget, ha
$\{ B1, B2, ...Bk \} \subseteq \{A1, A2, ...An\}$ **triviálisnak** nevezzük (mert ennek minden reláció eleget tesz).

2. **Bővíthető**:Ha $\alpha \rightarrow \beta$, akkor $\alpha\gamma \rightarrow \beta\gamma$

3. **Tranzitív:**Ha $\alpha \rightarrow \beta$ és $\beta \rightarrow \gamma$ akkor $\alpha \rightarrow \gamma$

Bizonyítás:

1. Ha $t_1[\alpha] = t_2[\alpha]$, akkor $t_1[\beta] = t_2[\beta]$ teljesül, hiszen $\beta \subseteq \alpha$

2. A γ hozzávétele a bal oldalhoz azt jelenti, hogy az eddigi, α-n egyező sorokból csak azokat tekintjük, amelyek γ-n is egyeznek. Ezek a sorok tehát egyeznek β-n is és γ-n is.

3. $\alpha \rightarrow \beta$ miatt: $t_1[\alpha] = t_2[\alpha]$ esetén $t_1[\beta] = t_2[\beta]$
$\beta \rightarrow \gamma$ miatt: $t_1[\beta] = t_2[\beta]$ esetén $t_1[\gamma] = t_2[\gamma]$

Következmény: minden $t_1[\alpha] = t_2[\alpha]$ esetén $t_1[\gamma] = t_2[\gamma]$ fennáll, ami a definíció szerint éppen $\alpha \rightarrow \gamma$.

Definíció: Azt mondjuk, hogy az F funkcionális függőségi halmaz **logikai következménye** az $\alpha \rightarrow \beta$ funkcionális függőség, ha minden F-nek eleget tevő reláció előfordulás eleget tesz az $\alpha \rightarrow \beta$ funkcionális függésnek is.

Tétel: Az Armstrong axiómák helyesek és teljesek

Helyesség: Ha az adott F függőségi halmazból az axiómák felhasználásával az $\alpha \rightarrow \beta$ funkcionális függőség levezethető, akkor $\alpha \rightarrow \beta$ az F logikai következménye.

Bizonyítás: ld. fent

Teljesség: Ha egy funkcionális függőség F-nek logikai következménye, akkor az F-ből az Armstrong axiómák felhasználásával levezethető.

Bizonyítás nélkül.

Definíció: Az F funkcionális **függőségi halmaz lezártja F^+** azokból a funkcionális függőségekből áll, melyek F-nek logikai következményei. (Praktikusan: F-ből az Armstrong axiómák segítségével levezethetők-hiszen az axiómák helyesek)

További tulajdonságok

Összevonhatósági szabály:

Ha
$A_1, A_2, ...A_n \rightarrow B_1$
$A_1, A_2, ...A_n \rightarrow B_2$
.
.
$A_1, A_2, ...A_n \rightarrow B_k$

funkcionális függőségek mindegyike igaz az adott relációra, akkor a definícióból közvetlenül adódik, hogy

$A_1, A_2, ...A_n \rightarrow B_1, B_2, ...B_k$.

Szétvághatósági szabály:

Ha $A_1, A_2, ...A_n \rightarrow B_1, B_2, ...B_k$, akkor

$A_1, A_2, ...A_n \rightarrow B_1$
$A_1, A_2, ...A_n \rightarrow B_2$
.
.
$A_1, A_2, ...A_n \rightarrow B_k$

Példák formális levezetésre

I. **Pszeudotranzitivitás:**

Igazoljuk, hogyha $\alpha \rightarrow \beta$ és $\beta\gamma \rightarrow \delta$, akkor $\alpha\gamma \rightarrow \delta$.

Bizonyítás:
1.
$\alpha \rightarrow \beta$ -ból γ-val való kibővítéssel : $\alpha\gamma \rightarrow \beta\gamma$
2.
$\alpha\gamma \rightarrow \beta\gamma$ és $\beta\gamma \rightarrow \delta$ -ból a tranzitivitási szabállyal: $\alpha\gamma \rightarrow \delta$.

II.
Ha $\alpha \rightarrow \beta$ és $\gamma \rightarrow \delta$, akkor $\alpha\gamma \rightarrow \beta\delta$.

Bizonyítás:

1.
$\alpha \rightarrow \beta$-t bővítsük γ-val: $\alpha\gamma \rightarrow \beta\gamma$
2.
$\gamma \rightarrow \delta$-t bővítsük β-val: $\gamma\beta \rightarrow \delta\beta$
3.
$\alpha\gamma \rightarrow \beta\gamma$ és $\gamma\beta \rightarrow \delta\beta$ -ból a tranzitivitás miatt következik $\alpha\gamma \rightarrow \beta\delta$

Megjegyzések

1. Valamely relációelőfordulás mindig kielégíti az $\alpha \rightarrow \beta$ függőséget, ha nincsenek olyan sorai, amelyek az α-n megegyeznek. Vagyis, ha egy előfordulásban minden érték különböző, akkor ez az előfordulás minden funkcionális függésnek megfelel.

2. A definícióban az R reláció **sémájának** hangsúlyozása igen fontos, hiszen a funkcionális függőségek teljesülése nem az adott reláció előfordulásaitól függ, hanem a reláció sémájára előírt tulajdonság, amely az attribútumok kapcsolatait fejezi ki, függetlenül az adott előfordulástól. E leíráson keresztül az adatstruktúrára, és így az egész adatbázis tervezésére vonatkozó fontos ismereteket rögzít.

Relációk kulcsai

Definíció: Az {A_1, A_2, ...A_n} attribútumhalmaz az R (A_1, A_2, ...A_n, B_1, B_2, ...B_k) sémájú reláció **kulcsjelöltje**, ha
1.) A_1, A_2, ...$A_n \rightarrow B_1$, B_2, ...B_k fennáll, és

2.) nincsen olyan **valódi** részhalmaza az { A_1, A_2, ...A_n } halmaznak, amely az R reláció összes többi attribútumát funkcionálisan meghatározná.

A kulcsjelölt tehát a reláció egyes sorainak azonosítására szolgál, hiszen minden sor különbözik az A_1, A_2, ...A_n attribútumokon. Ha nem így lenne, akkor a funkcionális függés miatt két egyforma sor szerepelne a relációban, amely ellentmond a reláció halmazelméleti definíciójának.

Egy relációnak több kulcsjelöltje is lehet.

A kulcsjelöltek közül ki kell jelölni egyet, amelyet ténylegesen használni akarunk a sorok azonosítására. A továbbiakban ezt nevezzük **elsődleges kulcs**nak. A kiválasztás az alapján történik, melyik kulcsjelölt áll a legkevesebb attribútumból. Amennyiben több olyan kulcsjelölt is van, melynek attribútumszáma azonos, azt kell kiválasztani, melynek adattárolásai igénye kisebb. pl. egy numerikus adat, melynek hossza 9 számjegy, vagy egy karakteres, mely 30 karakter hosszúságú.

Jelölés: A reláció sémájában aláhúzzuk azokat az attribútumokat, amelyek az elsődleges kulcs elemei.

A **kulcsjelölt egyszerű**, ha egyetlen attribútumból áll. A nem egyszerű kulcs **összetett**.

Definíció: A reláció kulcsjelöltjeinek elemeit **elsődleges attribútumok**nak, a reláció többi attribútumát **másodlagos attribútum**nak nevezzük.

A kulcsjelölt definíciójában a 2. tulajdonság a kulcsot alkotó attribútumhalmaz számosságának minimalitását fejezi ki. Nyilvánvaló, hogy bármely további attribútumot a kulcsjelöltet alkotó attribútumok halmazához hozzávéve az így kapott attribútumhalmaz szintén azonosítja a sorokat. Ezért a következő elnevezést használjuk:

Definíció: Valamely kulcsjelöltet tartalmazó bármely attribútumhalmazt **szuperkulcs**nak nevezzük.

Definíció: Azt az SK attribútumhalmazt, amelyre SK $\rightarrow$R teljesül, az R reláció **szuperkulcsá**nak nevezzük.

Megjegyzés: A magyar nyelv alapértelmezésével ellentétben tehát a szuperkulcs éppen nem a legjobb kulcsot jelenti. Az elnevezés oka az angol superset szó, amellyel valamely halmazt (set) tartalmazó halmazt neveznek (superset).

Példák:

1.A Nyilvántartás (TAJ-szám, név, cím, fizetés) relációban, amint az aláhúzás mutatja, a TAJ-szám a kulcs.

Szuperkulcsokra példa:

{ TAJ-szám, név},
{TAJ-szám, cím, fizetés},
{TAJ-szám, név, cím, fizetés}.

2. Film2(cím, év, hossz, szalagfajta, stúdióNév, stúdióCím)

3. Film(cím, év, hossz, szalagfajta, stúdióNév, színészNév)

Attribútumhalmazok lezárása

Attribútumhalmazok lezárása adott S függőségi halmaz szerint:
$\{A_1, A_2, ... A_n\}^+ = \{B | \exists A_1, A_2, ... A_n \rightarrow B$, amely S-ből következik$\}$

Attribútumhalmaz lezárásának kiszámítása

Bemenet: X attribútumhalmaz, F függőségi halmaz
Eredmény: X attribútumhalmaz F szerinti lezártja, X^+

FONTOS: F+-szal kell számolni!

Módszer:

1. X(0) = X
2. X(i+1) = X(i) U A, ha F^+-ban van olyan függőség, hogy $Y \rightarrow Z$, ahol $A \subseteq Z$, és $Y \subseteq X(i)$
3. Kilépés: X(i+1) = X(i)

Tétel: Ez a módszer helyesen számolja ki X^+-t

Bizonyítás: Teljes indukcióval

Mire lehet használni az attribútumhalmaz lezártját?

1. Kulcs megkeresésére:

$\{\alpha\}$ kulcs, ha $\{\alpha\}^+$ az összes attribútum, de A egy valódi részhalmazára sem igaz ez.

2. Függőség következménye-e az adott függőségi halmaznak (pl. normálformák ellenőrzésénél):

$\alpha \rightarrow \beta$ akkor és csak akkor következménye F-nek, ha $\beta \in \alpha^+$

Példák:

1. **Határozzuk meg a reláció kulcsait**, ha sémája R(A, B, C, D), és a hozzátartozó függőségi halmaz F:={AB→C, C→D, D→A}

2. **Határozzuk meg a reláció kulcsait**, ha sémája R(A, B, C, D), és a hozzátartozó függőségi halmaz F:={AB→C, C→D, D→A}

Megoldás: Figyelmesen végignézve a funkcionális függőségek bal oldalait, az a sejtésünk, hogy DB szuperkulcs:
1. $DB^+_1 = \{D, B\}$ a reflexivitás miatt
2. $DB^+_2 = \{D, B\} \cup \{A\}$ a D→A függőség miatt
3. $DB^+_3 = \{D, B, A\} \cup \{C\}$ a AB→C függőség miatt
4. $DB^+_4 = \{D, B, A, C\}$
5. $DB^+_4 = DB^+_5$

DB kulcsjelölt, ugyanis a minimalitási feltétel is teljesül: $D^+ = \{D\}$, $B^+ = \{B\}$

Másik kulcsjelölt a CB, mivel
CB^+={C, B, D, A}, D a C→D, A pedig a D→A függőség miatt. Mivel C^+={C, D, A}, B^+={B}, a minimalitás is teljesül.

Fentiekhez hasonlóan belátható, hogy AB is kulcsjelölt.

3. Befektetési irodai példa

Séma:
Attribútumok:
{Bróker, Iroda, Befektető, Részvény, Darab, Osztalék}
Függőségi halmaz=
{Bróker → Iroda,
Részvény→ Osztalék,
Befektető, Részvény → Darab
Befektető → Bróker}

Határozzuk meg a reláció kulcsát az előző megoldások mintájára!

4. Következménye-e a C→A, illetve a D→C függőség az F:={AB→C, C→D, D→A} függőségi halmaznak?

Megoldás:

C^+={C, D, A}, így A∈ C^+, tehát C→A következik F-ből.

D^+={ D, A} így C∉ D^+, tehát D→C nem következik F-ből.

Függőségi halmaz lezártja

Definíció: Az F funkcionális **függőségi halmaz lezártja F^+** azokból a funkcionális függőségekből áll, melyek F-nek logikai következményei. (Praktikusan: F-ből az Armstrong axiómák segítségével levezethetők-hiszen az axiómák helyesek)

$F^+=\{\alpha\rightarrow\beta \mid \alpha\rightarrow\beta$ F-ből levezethető$\}$

Példa:

R (A, B, C, G, H, I)

$F=\{A\rightarrow B, A\rightarrow C, CG\rightarrow H, CG\rightarrow I, B\rightarrow H\}$

$F^+=\{A \rightarrow A, B\rightarrow B, \ldots\ldots AB\rightarrow AB, \ldots..A\rightarrow BC, A\rightarrow H, AG\rightarrow I, CG\rightarrow HI \ldots\ldots\}$

Függőségi halmazok lezárása

$F^+:=\{X\rightarrow Y \mid X\rightarrow Y$ következménye F-nek $\}$

Következmény eldöntése: X^+ segítségével

Bázis: Lezártja tartalmazza F összes következményét

Minimális bázis: nincs olyan valódi részhalmaza, amelynek lezártja F összes következményét tartalmazná.

Relációk felbontása

Adatbázis ideális tulajdonságait ún. normálformákban rögzítjük. Az E/K modell relációkba való átírásakor ezek nem feltétlenül teljesülnek. Azt a folyamatot, amelynek során átalakítjuk az adatbázist a kívánt normálformára, **normalizálás**nak nevezzük.

A normalizálás célja a hibaforrást jelentő ún. **anomáliák minimálisra** csökkentése.

Az anomáliák oka: legtöbbször adatok többszörös, **redundáns** tárolása

Cél: a redundancia csökkentése, ellenőrzött redundacia

Módszer: felbontás (dekomponálás)

Anomáliák

Változtatási anomáliának nevezzük a
- módosítási,
- beírási
- törlési

anomáliákat.

Módosítási anomália: Egy attribútum értékének megváltozását több helyen kell kijavítani. Ez az adatbázis **inkonzisztens** állapotához vezethet.

Beírási anomália: Új egyedelőfordulásra vonatkozó információ bevitele nem lehetséges hiányos adatok miatt.
Ez információvesztést eredményez.

Törlési anomália: Bizonyos adatok törlésével elvesznek olyan adatok (mivel csak egész sorokat tudunk törölni), amelyekre továbbra is szükség lenne az adatbázis-kezelő rendszer használatában. Ez is információvesztést eredményez.

Példa
Milyen anomáliák lépnek fel az
Oktatók(TanárAzonosító, TanárNév, Cím, Telefon, TantárgyNév, Óraszám, Követelmény)
relációval kapcsolatban?

Redundancia:
Hányszor szerepelnek az oktatók adatai?
A tantárgyak adatai?
A (z ellenőrizetlen) redundancia anomáliákhoz vezet.

Módosítási anomália:
Mi történik, ha megváltozik egy oktató címe?

Törlési anomália:
Mely információk vesznek el, ha egy oktató kilép?

Beírási anomália:
Új tantervet vezetnek be, de még nincsen oktató, aki tanítani fogja. Mit eredményezne az új tantárgy felvétele az adatbázisba? (NULL-értékeket)
Mi az oka, és hogyan lehet elkerülni a példában található anomáliákat?
Az Oktatók reláció funkcionális függőségi viszonyait látjuk az alábbi függőségi halmazban:
F={TanárAzonosító $\rightarrow$TanárNév, Cím , Telefon;
TantárgyNév $\rightarrow$ Óraszám, Követelmény;
TanárAzonosító $\rightarrow$ TantárgyNév}

Azt várhatjuk, hogy a relációnak a következő relációkra bontása megoldja a problémákat (ellenőrizzük!):
Oktatók(TanárAzonosító, TanárNév, Cím , Telefon)
Tantárgyak(TantárgyNév, Óraszám, Követelmény)
Tanít(TanárAzonosító, TantárgyNév)

Reláció felbontása (dekomponálása)

Definíció: Az $R(A_1, A_2, ...A_n)$ sémájú reláció egy felbontása az
$R_1(A_{11}, A_{12}, ...A_{1i})$,
$R_2(A_{21}, A_{22}, ...A_{2j})$,
:
$R_k (A_{k1}, A_{k2}, ...A_{kn})$

relációk halmaza, ha:
$U A_{nm} = \{ A_1, A_2, ...A_n \}$

R_k egy előfordulása:= $_{Ak1, Ak2, ...Akn}(r)$
ahol r az R sémának megfelelő előfordulás.

A felbontás az anomáliák elkerüléséhez szükséges

Normalizálás

A normalizálás során a kiinduló relációt felbontjuk/dekomponáljuk több relációra, melynek során a bővítési, módosítási és törlési anomáliák megszüntetése a cél a redundáns adattárolás minimlizálása mellett.

Adatbázistervezési feladatok megoldásához szükséges normálformák:
1-3NF, BCNF: funkcionális függőségek
4NF: funkcionális és többértékű függőségek
5NF: join függőségek

A könyvben az 1-3 NF-el és a BCNF-el fogunk foglalkozni. Ezek ismerete már elegendő egy nem túl összetett adatbázisfeladat megfelelő megoldásához.

A normalizálás után az adatbázisban továbbra is megtalálható bizonyos fokú redundancia, de ez ellenőrzött formában, ezért nem vezet anomáliához.

Első normálforma

Definíció: Első normálforma (1NF): A reláció minden sorában pontosan egy elemi attribútumérték áll. A reláció matematikai definíciója miatt a relációban nem lehetnek azonos sorok, ezért az 1NF-ben lévő relációnak van kulcsa, továbbá, az 1NF definíciójából következően minden (nem kulcs) attribútum funkcionálisan függ a kulcstól.

Informálisan az R reláció 1. normálformában, 1NF-ben van, ha minden sorának minden attribútumértéke **elemi** és **egy értéket** vesz fel.

A nem elemi **attribútum**ot új attribútumok beiktatásával lehet megszüntetni.
A **többértékűség (halmazérték)** megszüntetése úgy történik, hogy minden sort annyiszor leírunk, ahányszor szükséges.

Egy általánosabb definíciója az első normál formának:
Egy reláció első normálformában van, ha nem tartalmaz teljesen azonos sorokat, minden attribútum csak egy értéket vehet fel (pl. a dátum helyére csak egy dátumot írunk) és az attribútumok sorrendje minden sorban azonos.

Példa:
A Hallgató(TanárAzonosító, TanárNév, **Cím**, **TantárgyNév**, Jegy) sémájú reláció nem 1NF-ben van, mert a **Cím** attribútum nem elemi: a város, utca, házszám tulajdonságokból áll. A **Tantárgy** attribútum pedig több értéket vesz fel, halmazértékű.

Megjegyzés: A legtöbb, kereskedelemben kapható rendszer rendelkezik ún. objektum-orientált bővítéssel, amely lehetővé teszi a reláció beágyazását, illetve halmazérték kezelését.

Példa(ismétlés): Láttuk, hogy az
Oktatók(TanárAzonosító, TanárNév, Cím, Telefon, TantárgyNév, Óraszám, Követelmény)
relációnál az anomáliák mindegyike fellép; módosítási, törlési, beírási. Az anomáliákat felbontással kerültük el.

Mi volt az oka az anomáliáknak?

- két egyedhalmaz került egy relációba, így a megfelelő kulcsok funkcionálisan meghatározzák a hozzákapcsolódó attribútumokat
- másképpen: egyes attribútumokat a kulcs egy részhalmaza már funkcionálisan meghatároz, és okozza a redundanciát.

Második normálforma

Az Oktatók(TanárAzonosító, TanárNév, Cím, Telefon, TantárgyNév, Óraszám, Követelmény) reláció
F={TanárAzonosító $\rightarrow$TanárNév, Cím , Telefon;
TantárgyNév $\rightarrow$ Óraszám, Követelmény;
TanárAzonosító $\rightarrow$ TantárgyNév}

alábbi felbontása valóban megszünteti az anomáliákat:

Oktatók(TanárAzonosító, TanárNév, Cím , Telefon)
Tantárgyak(TantárgyNév, Óraszám, Követelmény)
Tanít(TanárAzonosító, TantárgyNév)

Definíció: Azt mondjuk, hogy a **reláció második normálformában** (2NF) van, ha minden másodlagos attribútum funkcionálisan teljesen függ a reláció bármely kulcsától.

Azt mondjuk, hogy az A $\rightarrow$ B funkcionális függőség **sérti** a normálformát, ha az nem felel meg a normálforma definíciójában előírtaknak.

Az előző példában az anomáliákat az okozta, hogy egyes attribútumok funkcionális függése a kulcstól részleges (nem teljes) volt.

Egy általánosabb definíciója a második normál formának:
Egy reláció második normálformában van, ha első normálformában van (tehát teljesíti az első normálforma követelményeit) és minden nem kulcs attribútum a kulcstól függ funkcionálisan, annak részétől nem (tehát összetett kulcs esetén a kulcs minden eleme szükséges a funkcionális függéshez)

Módszer a második normálformára bontáshoz:
Veszteségmentes, adott normálformában (1-4NF, BCNF) levő relációkat kapunk a következő felbontással:
Az eredeti R relációt két másik relációra bontjuk a sértő függőség szerint:

1. R_1 sémája
2. R_2 sémája R^-

A kapott R_1 és R_2 relációk közül R_1 általában a kívánt normálformában van, R_2-t meg kell vizsgálni, és ha szükséges, folytatni az eljárást.

Megjegyzés:

Szándékosan függőséget írtunk funkcionális függőség helyett, mert az a módszer más függőségek, pl. többértékűek esetében is célravezető, 4NF-re hozásnál.

Példa 2NF-re hozásra:
Oktatók(TanárAzonosító, TanárNév, Cím, Telefon, TantárgyNév, Óraszám, Követelmény)
F={TanárAzonosító $\rightarrow$TanárNév, Cím , Telefon;
TantárgyNév $\rightarrow$ Óraszám, Követelmény;
TanárAzonosító $\rightarrow$ TantárgyNév}

Felbontással:
A kulcstól való részleges funkcionális függésben levő attribútumokkal új sémát készítünk (ezen funkcionális függőségek sértik a 2NF-et, hiszen részleges a függés):

A sértő: _____TanárAzonosító $\rightarrow$TanárNév, Cím , Telefon)
R_1 sémája Oktatók$_1$ (TanárAzonosító, TanárNév, Cím , Telefon)
R_2 sémája R- Oktatók$_2$ (TanárAzonosító, TantárgyNév, Óraszám, Követelmény)

R_2 nincsen 2NF-ben, mert TantárgyNév$\rightarrow$Óraszám, Követelmény (részleges függés)

Ezért: R3: Tantárgyak(TantárgyNév, Óraszám, Követelmény)
R4: $R2^-$ Tanítja (TanárAzonosító, TantárgyNév)

A dekompozíció eredménye: R1, R3, R4. E relációk mindegyike 2NF-ben van.

Harmadik normálforma (3NF)

Definíció: A γ attribútumhalmaz **tranzitívan függ** az α attribútumhalmaztól, ha létezik egy olyan β attribútumhalmaz, amelyet α funkcionálisan meghatároz, és amelytől γ funkcionálisan függ: $\alpha \rightarrow \beta$ és $\beta \rightarrow \gamma$.

Példa:

Tekintsük az alábbbi sémával megadott relációt:
Alkalmazottak(DolgozóAzonosító, DolgozóNév, DolgozóCím, DolgozóTelefon, RészlegNév, Fizetés, Telephely, RészlegCím, RészlegTelefon)

Első vagy második normálformában van-e a reláció?
Írjuk fel az összes funkcionális függőséget!
Keressük meg a tranzitív függéseket!
Milyen anomáliákat tapasztalunk?
Mi az oka az anomáliáknak?

Harmadik normálforma (3NF)

Definíció1: A **reláció harmadik normálformában (3NF)** van, ha 2NF-ben van, és egy másodlagos attribútum sem függ tranzitívan a reláció egyetlen kulcsától sem.

Definíció2: A **reláció harmadik normálformában** (3NF) van, ha minden funkcionális függőségre vagy szuperkulcs, vagy elsődleges attribútum.

Tétel: Definíció1 ekvivalens Definíció2-vel

Bizonyítás:
I.
Definíció2-ből következik Definíció1.
Tegyük fel, hogy sérti a 3NF-et Definíció2 szerint.

Ez akkor lehetséges ha:
nem szuperkulcs és nem elsődleges

Ha nem szuperkulcs, akkor vagy
Kulcs, ezért részleges függést okoz az (nincs 2NF-ben a reláció)
Kulcs, akkor, mivel Kulcs teljesül, így -val Kulcs tranzitív függésben van.

II.
Definíció1-ből következik Definíció2:
Ezt a részt nem bizonyítjuk.

Az Alkalmazottak(DolgozóAzonosító, DolgozóNév, DolgozóCím, DolgozóTelefon, RészlegNév, Fizetés, Telephely, RészlegCím, RészlegTelefon)

reláció 2NF-ben van, hiszen a kulcs **egyszerű**.

Nincsen 3NF-ben, mert a RészlegNév funkcionálisan meghatározza a Telephelyet, a RészlegCím-et, és a RészlegTelefon-t, így a Telephely, RészlegCím, RészlegTelefon tranzitívan függ a kulcstól.

3NF-re hozás:

A
RészlegNév → Telephely, RészlegCím, RészlegTelefon
funkcionális függés sérti a 3NF-et, ezért:

R_1= Részleg (RészlegNév, Telephely, RészlegCím, RészlegTelefon)

R_2= Alkalmazottak(DolgozóAzonosító, DolgozóNév, DolgozóCím, DolgozóTelefon, RészlegNév, Fizetés)

Összefoglaló példa az 1-3NF-re hozásra
(Házi feladat1: miként módosul az alábbi példa, ha Rendelésszámot is felveszünk a relációba?)
(Házi feladat2: miként módosul az alábbi példa, ha BCNF-re akarjuk hozni?):

A Rendelés reláció sémája a következő:

Rendelés(VevőKód, VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, SzámlaSorszám, ÁruKód, ÁruNév, Egységár, Mennyiség, RendelésDátuma, FizetésDátuma, FizetésiMód).

A relációhoz tartozó függőségi halmaz:

F={ VevőKód → VevőNév, VevőIrányítószám, VevőVáros, VevőUtca;
VevőIrányítószám →VevőVáros;
ÁruKód → ÁruNév, Egységár;
SzámlaSorszám → VevőKód, VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, RendelésDátuma, FizetésDátuma, FizetésiMód;
SzámlaSorszám ÁruKód → Mennyiség}

Gondoljuk végig a funkcionális függéseket. A VevőKód a vevő adatait, az ÁruKód az áru adatait tudja meghatározni (kivéve, mekkora mennyiséget vett valaki, hiszen ugyanabból az áruból akár más-más mennyiséget vásárolhatnak a vevők), a SzámlaSorszám szinte minden adatot (ki, vásárolt, mikor, milyen módon fizetett, az árut és mennyiség viszont nem, hiszen egy megrendelésnél több féle árut vehetünk), a mennyiséget pedig csak SzámlaSorszám és az adott vásárláshoz tartozó ÁruKód határozza meg egyértelműen.

1NF: Gondoljuk végig, 1NF-ben van-e a reláció?
Ami segít, hogy megnézzük, hogy néznek ki az adatok beírva egy táblázatba, mellyel a relációs adatbáziskezelők dolgoznak.
Ha egy rendelés első tételét rögzítjük, akkor felveszünk minden adatot. Amikor a második árut rögzítjük ehhez a rendeléshez, akkor a SzámlaSorszám és a vevőadatok, fizetéssel kapcsolatos adatok ismétlődnek (bővítési anomália), de az áru adatai eltérőek lesznek. Így teljesül az 1NF kritériuma, miszerint nincsenek teljesen azonos sorok, egy attribútum, csak egy értéket vehet fel az az attribútumok sorrendje azonos. Ha egy vevőnk címe megváltozik, minden sorban, ahol az ő vásárlásai szerepelnek, módosítanunk kellene a cím adatokat

(módosítási anomália), ha pedig egy vevő kéri, hogy töröljük az adatait, minden érintett sort törölnünk kellene, de így elvesznének a vásárlási adatai is (törlési anomália)

Tehát ebben a normálformában nem tudjuk az anomáliákat megszüntetni és még redundáns adattárolásunk is van bőven, hiszen ha a egy másik vevő ugyanazt az árut vásárolja, az áru adatok is ismétlődnek. Nem hagyhatjuk így, dekomponálnunk/normalizálnunk kell a relációt.

Emellett, ahogy a bevezetőben is említettük, nem mindig van lehetőségünk a legjobb megoldást elkészíteni, erre most rögtön látunk két esetet.
Az egyik a vásárlásnál az áruk adatainak rögzítési módja a kerékkötőnk. Gondoljunk csak bele, amikor olyan helyen vásárolunk, ahol futószalagra tesszük fel az árut és azonos cikkeket is vásárolunk, de a sorrendben nem követik egymást. Ilyenkor a blokkon rögtön rögzítésre kerül minden lehúzott termék, tehát nem egy összesített darabszámot látunk, hanem a tételeket külön, de azonos blokksorszámmal. Ilyen esetben még az első normálforma követelménye sem teljesül 0 NF-ben lesznek az adatok rögzítve.

A másik esetet ebben a példának a végén beszéljük meg.

Amennyiben pl. webáruházban válogattuk össze a termékeket ott maga a rendszer nem engedi, hogy az egyes árutételek külön legyenek a számlára felvéve, hanem az áru mellett a mennyiség jelzi, hány darabot is vásároltunk.

A fenti függőségi halmazból következik, hogy a reláció kulcsa összetett: SzámlaSorszám ÁruKód.

Az előbb már megbeszéltük, hogy ilyen esetben 1NF-ben van a relációnk.

2NF: A reláció viszont nincsen 2NF-ben. Miért?

A 2NF definíciója szerint minden nem kulcs attribútum csak a kulcstól függhet, annak részétől nem.

A kulcs részhalmazától való függés található az Áru adatok esetében, mert az ÁruKód funkcionálisan meghatározza a következő attribútumokat:
ÁruNév, Egységár

Tehát az ebben a funkcionális függőségben található attribútumhalmazokból képezünk egy sémát:

Áru(<u>ÁruKód</u> ÁruNév, Egységár)

és érvényes a relációban az alábbi funkcionális függés:
$F_{áru}$={ ÁruKód → ÁruNév, Egységár}

a kulcsa pedig az ÁruKód

A rendelés táblában marad a többi adat és az ÁruKód, hiszen a kulcs része:
Rendelés(VevőKód, VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, <u>SzámlaSorszám</u>, <u>ÁruKód</u>, Mennyiség, RendelésDátuma, FizetésDátuma, FizetésiMód).

A relációhoz tartozó függőségi halmaz:

F={ VevőKód → VevőNév, VevőIrányítószám, VevőVáros, VevőUtca;
VevőIrányítószám →VevőVáros;
SzámlaSorszám → VevőKód, VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, RendelésDátuma, FizetésDátuma, FizetésiMód;
SzámlaSorszám ÁruKód → Mennyiség}

kikerült a függőségi halmazból a ÁruKód → ÁruNév, Egységár funkcionális függés, mert **egy függés csak akkor van érvényben egy relációban, ha mind a bal, mind a jobb oldalán található attribútumok szerepelnek a relációban**, de az ÁruNév, Egységár mezők már kikerültek ebből a relációból.

Most már 2NF-ben vannak a relációk?
Az áru reláció 2NF-ben van, hiszen a kulcs egy attribútumból áll, tehát nincs részhalmaza, és 3NF-ben is, mivel nem tartalmaz tranzitív függést.

A rendelés reláció viszont nincs 2NF-ben, mivel pl. a SzámlaSorszámtól tehát a kulcs részhalmazától való funkcionális függést tartalmaz.

Ehhez a funkcionális függéshez tartozó adatokból képezünk külön sémát (kulcs a SzámlaSorszáma lesz):

Számla(SzámlaSorszám, VevőKód, VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, RendelésDátuma, FizetésDátuma, FizetésiMód)

A relációhoz tartozó függőségi halmaz:

$F_{számla}$={ VevőKód → VevőNév, VevőIrányítószám, VevőVáros, VevőUtca;
VevőIrányítószám →VevőVáros;
SzámlaSorszám → VevőKód, VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, RendelésDátuma, FizetésDátuma, FizetésiMód;

A rendelés táblában a dekomponálás után már csak a következő adatok maradnak:
Rendelés(SzámlaSorszám, ÁruKód, Mennyiség)

A relációhoz tartozó függőségi halmaz:

F=(SzámlaSorszám ÁruKód → Mennyiség}

Most már 2NF-ben vanna a relációk?
A rendelés reláció igen, hiszen egy funkcionális függést tartalmaz és a Mennyiség értéke csak a kulcs egészétől függ. 3NF-ben is van pont azért, mert csak egy funkcionális függést tartalmaz, a tranzitivitáshoz pedig kettő kellene.

A számla reláció viszont nincsen 3NF-ben, mert több tranzitív függést is tartalmaz.
Nézzük az elsőt:
A VevőKód funkcionálisan meghatározza a VevőNév, VevőIrányítószám, VevőVáros, VevőUtca, viszont megnézve a függőségi halmazt, a VevőIrányítószám is meghatározza a VevőVáros mező értékét:

VevőKód → VevőIrányítószám; VevőIrányítószám → VevőVáros

Ilyen esetben a tranzitív függés második elemével képzünk új sémát:
Irszám(VevőIrányítószám, VevőVáros)
$F_{irszám}$={VevőIrányítószám → VevőVáros}

maradt a rendelés az alábbi séma szerint:

Számla(VevőKód, VevőNév, VevőIrányítószám, VevőUtca, SzámlaSorszám, RendelésDátuma, FizetésDátuma, FizetésiMód).

A relációhoz tartozó függőségi halmaz:

F={ VevőKód → VevőNév, VevőIrányítószám, VevőUtca;
SzámlaSorszám → VevőKód, VevőNév, VevőVáros, VevőUtca, RendelésDátuma, FizetésDátuma, FizetésiMód;}

Mit értünk el ezzel a felbontással?
Biztos mindenki találkozott a pandémia alatti webes rendeléseinél olyan esettel, amikor beírta az irányítószámát és a felületen megjelent a város neve, amit így nem kellett/lehetett kitölteni. Ilyenkor a rendszerben van egy Irszám reláció, mit amit mi a tranzitív függés feloldásával létrehoztunk és feltöltik az adott országban található adatokkal, így azonos irányítószámról beérkező rendelések esetén csak az irányítószám értéke tárolódik redundánsan, a város értéke már nem.
Az Irszám reláció 2NF-ben és 3NF-ben is van. Kulcs a VevőIrányitószám, egy attribútumból áll a kulcs, tehát nincs részhalmaza (2NF) és csak egy funkcionális függést tartalmaz (3NF).

Keressük meg a másik tranzitív függést a Számla relációban!

A SzámlaSorszám funkcionálisan meghatározza a Vevőkód értékét, a Vevőkód értéke pedig a VevőNév, VevőIrányítószám, VevőUtca mezők értékét, így tranzitív függésben vannak az utóbbi attribútumok a SzámaSorszámtól. Ezért ezekkel az attribútumokkal készítünk egy relációsémát:

Vevő(VevőKód, VevőNév, VevőIrányítószám, VevőUtca)

A relációhoz tartozó függőségi halmaz:

$F_{vevő}$={ VevőKód → VevőNév, VevőIrányítószám, VevőUtca}

Nézzük mely attribútumok maradtak a felbontás után a rendelés relációban:

Számla(VevőKód, SzámlaSorszám, RendelésDátuma, FizetésDátuma, FizetésiMód).

A relációhoz tartozó függőségi halmaz:

F={SzámlaSorszám → VevőKód, RendelésDátuma, FizetésDátuma, FizetésiMód}

Mit értünk el ezzel a felbontással?
A vevő adatai külön relációba kerültek, így ha a vevőnk többször vásárol nálunk, az adatai csak egyszer lesznek letárolva, redundánsan csak a VevőKód tárolódik el. A Vevő reláció 2NF-ben és 3NF-ben is van. Kulcs a VevőKód, egy attribútumból áll a kulcs, tehát nincs részhalmaza (2NF) és csak egy funkcionális függést tartalmaz (3NF).

A Számla reláció is 2NF-ben és 3NF-ben van. Kulcs a SzámlaSorszám, egy attribútumból áll a kulcs, tehát nincs részhalmaza (2NF) és csak egy funkcionális függést tartalmaz (3NF).

A kiinduló rendelés relációt tehát az alábbi 3NF-ben lévő relációkra bontottuk:

Áru(ÁruKód ÁruNév, Egységár)
Irszám(VevőIrányítószám, VevőVáros)
Vevő(VevőKód, VevőNév, VevőIrányítószám, VevőUtca)
Számla(VevőKód, SzámlaSorszám, RendelésDátuma, FizetésDátuma, FizetésiMód).
Rendelés(SzámlaSorszám, ÁruKód, Mennyiség)

Ez a felbontás néhány adattal kiegészítve (pl. rendelésszám) már alkalmas lehet egy Webáruház kiszolgálásához (ideális esetben).

Miért nem elfogadható ez a változat?

Korábban említettük, hogy lesz még egy eset, amikor nem tudjuk az optimális redundanciamentességet használni. Most a törvényi szabályozás szól közbe.
Képzeljük el, hogy a fenti relációs sémákat alkalmazva készítjük el a webáruházunkat. Szépen futnak be a rendelések, a cég jól prosperál az adott piaci környezetben, majd megjelennek az Adóhivatal munkatársai, hogy nyomtassunk ki egy 4 évvel ezelőtti rendeléshez tartozó számlát (Magyarországon pl. 5 évig visszamenőleg kell a rendszereknek ezt a kérést biztosítaniuk).
A rendszerünk alkalmas a számla kinyomtatására **DE!** az egységárból számított végösszeg nem biztos, hogy azonos lesz a 4 évvel ezelőttivel.

Miért?

Mert a rendszerünk az aktuális egységárral számol!

A felbontásunk jó, a lehető legkevesebb redundáns adatot tárolja le, mégsem felel meg a törvényi szabályozásnak.

Mit lehet tenni?

Két megoldás is kínálkozik.
Az egyik adattárolásban rosszabb, de a rendszer mögöttes működését szabályozó programkód egyszerűbb, a másik megoldás kevesebb redundanciát hoz magával az első megoldáshoz képest, de a mögöttes logika lesz bonyolultabb.

Megoldás I.

A legegyszerűbb megoldás, ha a rendelés táblát kiegészítjük egy attribútummal, az aktuális egységárral és a számlát a táblában található mennyiség és aktuális egységár adattal összeszorozva számoljuk, így bármikor lekérve az adott rendelés számláját, az akkori árral számolja a végösszeget, ami így mindig azonos lesz.

A módosított rendelés reláció:

Rendelés(SzámlaSorszám, ÁruKód, Mennyiség, **AktuálisEgységár**)

Megoldás II.

A másik megoldáshoz egy új táblát kell létrehoznunk, mely tartalmazza, az adott egységár meddig volt érvényben:

Árak(Árukód, Egységár, Dátum)

A számla összegének kiszámításához így minden vásárolt termék aktuális egységárát kell megkeresni és ez alapján számolni a végösszeget.

Az előző oldalakon levezetett összetett példa remélhetőleg érthetően mutatta be a normalizáció folyamatát és mutatott rá annak fontosságára a redundanciacsökkentés és hatékonyságnövelés területén.

Amikor egy adott feladathoz tartozó adatbázis sémáját kell megterveznünk az alábbi lépéseket érdemes végrehajtani:

1. *Az összes letárolandó attribútum összegyűjtése.*
2. *Az attribútumok közötti kapcsolatok felderítése.*
3. *A funkcionális függések meghatározása (ne legyen olyan attribútum, mely nem szerepel a funkcionális függőségi halmazban sem a jobb, sem a bal oldalon valamelyik elemnél, tehát nem határozza meg más attribútum értékét, vagy az ő értékét nem határozza meg egyik attribútum sem „lóg a levegőben”).*
4. *A funkcionális függések alapján határozzuk meg a kiinduló reláció kulcsát.*
5. *Végezzük el a normalizációt a törvényi szabályozás figyelembevételével.*

Boyce-Codd normálforma (BCNF)

Definíció: Az R reláció BCNF-ben van, ha minden nem triviális funkcionális függőség bal oldala szuperkulcs.

Feladat: Ellenőrizzük, hogy az előző (összefoglaló) példa végeredményeként adott relációk BCNF –ben vannak-e?

Példa:

R(A, B, C, D, E), a függőségi halmaz: F={A→D, B→E, D→C}.
Ellenőrizzük, hogy az R reláció BCNF-ben van-e.

Módszer:

Meg kell határoznunk a reláció kulcsát az attribútumok lezártjának megállapításával. Csak ezután tudjuk eldönteni a függőségekről, melyek bal oldala nem szuperkulcs.
Szuperkulcs: legalább a kulcs attribútumokat tartalmazza
Triviális függés: A→A, B→B, stb. A triviális függéseket nem szoktuk külön felsorolni a függőségi halmazban pont a trivialitásuk miatt.

A kulcs ismeretében az eredeti R relációt két másik relációra bontjuk a sértő függőség szerint:

R_1 sémája
R_2 sémája R^-

A kapott R_1 és R_2 relációk közül R_1 általában a kívánt normálformában van, R_2-t meg kell vizsgálni, és ha szükséges, folytatni az eljárást.

Kulcs meghatározás az attribútumok lezártjának segítségével:

$\{A\}^+=\{A, D, C\}$
$\{B\}^+=\{B, E\}$
$\{C\}^+=\{C\}$
$\{D\}^+=\{D, C\}$
$\{E\}^+=\{E\}$

Végig vettük az egy attribútumos lezártakat, de nem találtunk olyat, melynek lezártjában a reláció összes attribútuma szerepelne, ezért folytatjuk a két attribútumos lezártak meghatározásával a kulcskeresést (olyan sorrendben írjuk a lezártakba az attribútumokat, amilyen sorrendben meg tudtuk őket határozni a funkcionális függések alapján.

Kulcs meghatározás az attribútumok lezártjának segítségével (folytatás):
$\{AB\}^+=\{A, B, D, C, E\}$
$\{AC\}^+=\{A, D, C\}$
$\{AD\}^+=\{A, D, C\}$
$\{AE\}^+=\{A, E, D, C\}$
$\{BC\}^+=\{B, C, E\}$
$\{BD\}^+=\{B, D, E, C\}$
$\{BE\}^+=\{B, E\}$
$\{CD\}^+=\{C, D\}$
$\{CE\}^+=\{C, E\}$
$\{DE\}^+=\{D, E, C\}$

Az $\{AB\}^+$ tartalmazza egyedül a reláció összes attribútumát, ő a kulcsjelölt. Mivel a két attribútumos lezártak közül nincs másik megfelelő, már csak legközelebb a három attribútumos lezártak között találnák ilyet, így az AB attribútum a minimális a kulcsjelöltek közül (legkevesebb attribútumból áll), ezért ő lesz a reláció kulcsa:

R(<u>A</u>, <u>B</u>, C, D, E), a függőségi halmaz: $F=\{A\rightarrow D, B\rightarrow E, D\rightarrow C\}$.

Most már tudunk tovább dolgozni a BCNF meghatározásán.
Nézzük az első nem triviális funkcionális függést: $A\rightarrow D$
Mivel a bal oldalon szuperkulcsnak kellene lennie, tehát legalább az AB attribútumoknak, de csak az A attribútum szerepel ott, így egy a függés sérti a BCNF definícióját.
Ilyen esetben a Az eredeti R relációt két másik relációra bontjuk a sértő függőség szerint:
az egyik relációba kerül az érintett függésnek minden attribútuma, tehát

$R_1(A, D)$

a másik relációba az eredeti reláció minden attribútuma, kivéve az érintett függőség jobb oldalán lévő attribútumokat:

$R_2(A, B, C, E)$

Megnézzük, mely függések érvényesek az R_1 relációban:

$R_1(A, D)$
$F_1=\{A\rightarrow D\}$

A reláció kulcsa A (lezárt meghatározásával igazolható), egy érvényes nem triviális funkcionális függés van a függőségi halmazban, melynek bal oldalán megtalálható legalább a reláció kulcsa (a függés bal oldala szuperkulcs), ezért a reláció BCNF-ben van.

Megnézzük, mely függések érvényesek az R_2 relációban:

$R_2(A, B, C, E)$
$F_2=\{B\rightarrow E\}$

a $D\rightarrow C$ funkcionális függés nem érvényes, mert nincs benne a relációban minden attribútum a függésből!

A reláció kulcsa a ACE (lezárt meghatározásával igazolható), egy érvényes nem triviális funkcionális függés van a függőségi halmazban, melynek bal oldala nem szuperkulcs (nem tartalmazza legalább a kulcs attribútumokat, ezért a reláció nincs BCNF-ben!
Tovább bontjuk a sértő funkcionális függés mentén (a számozás első eleme utal arra a relációra, melyből tovább bontottuk):

R_{21}(B, E)
F_{21}={B→E}

R_{22}(A, B, C)
F_{22}={}

Az R_{21} BCNF-ben van, mert a funkcionális függés bal oldala szuperkulcs.

Meglepő, de

az R_{22} is BCNF-ben van, mert a funkcionális függőségi halmaz csak a triviális függéseket tartalmazza, így nem sérti a BCNF definícióját!

Példa:

R(A, B, C, D, E) felbontásakor egyik kapott reláció S(A, B, C), az adott függőségi halmaz:
F= F={A→D, B→E, DE→C}. Ellenőrizzük, S BCNF-ben van-e?

Ehhez tudni kell, mely függőségek teljesülnek S-ben F szerint?

$\{A\}^+$= {A, D} a D, nincsen benne S-ben
$\{B\}^+$= {B, E} az E nincsen benne S-ben
$\{C\}^+$= {C}

$\{AB\}^+$ = {A, B, C, D, E} ABA, ABB, ABAB ABE, ABD , **ABC**
(Kulcsjelölt AB)
$\{AC\}^+$ = {A, C, D} Triviálisak +ACD, de D nincsen S-ben

$\{BC\}^+$ = { B, C, E} BCC, BCA, BCE de E nincsen S-ben

Az egyetlen nem triviális függőség amely benne van S-ben: **ABC**.
Mivel AB kulcs, ezért nem sérti a BCNF-et.

Eredeti reláció visszaállítása (információ visszanyerése)

Amikor befejeztük a normalizálás és felbontottuk a kiinduló relációt több kisebb relációra a redundancia csökkentése, illetve az anomáliák megszüntetésének érdekében, jó lenne tudni, hogy vissza tudjuk-e állítani az eredeti relációt, vissza tudjuk-e kapni az eredeti tartalmát.
Erre keressük a megoldást az alábbiakban.

Vázlatos elv:
Adott R(A, B, C)

R egy felbontása: R_1(A, B), R_2(B, C)

R visszaállítása: $R_1 \bowtie R_2$
(a természetes összekapcsolás / join művelettel történik)

Természetes összekapcsolás művelete:
A természetes összekapcsolás az egyik leggyakrabban használt relációs algebrai művelet az adatbáziskezelésben. Segítségével lehet a dekomponálás után az összetartozó adatokat külön táblákból megjeleníteni.
Előfeltétel: a természetes összekapcsolás művelete akkor végezhető el két reláción, ha van közös attribútumuk, mely adattípusban és hosszban is megegyezik. Tehát ha az egyik táblában az adat karakteres és 4 karakter hosszú, akkor a másik táblában nem lehet karakteres, de 5 hosszú, vagy numerikus, stb. A 80-as 90-es években használt adatbáziskezelő rendszerek még egy kritériumot megfogalmaztak: a közös attribútum neve is legyen azonos. Ma már ez nem feltétel, mégis vagy azonos, vagy hasonló nevű attribútumokat használunk, hogy könnyebben beazonosítható legyen, mely attribútumok felhasználásával lehet a természetes összekapcsolás műveletét elvégezni.

Művelet: azokat a sorokat kapcsolja össze a két relációban, melyek értéke közös attribútumban megegyezik. **A közös attribútum csak egyszer jelenik meg az eredménytáblában.**

Példa:
R(Név, Irszám, Város), a benne tárolt adatok:

Név	Irszám	Város
Kovács Béla	3300	Eger
Horváth Miklós	1121	Budapest

dekomponálás után:
R_1(Név, Irszám),

Név	Irszám
Kovács Béla	3300
Horváth Miklós	1121

R_2(Irszám, Város)

Irszám	Város
3300	Eger
1121	Budapest

Szeretnénk visszaállítani a felbontás előtti relációt, így elvégezzük a természetes összekapcsolás műveletét:

$R_1 \bowtie R_2$

Név	Irszám	Város
Kovács Béla	3300	Eger
Horváth Miklós	1121	Budapest

Példa:
Adott R(A, B, C)

A	B	C
a	b	c
d	b	e

R egy felbontása: $R_1(A, B)$, $R_2(B, C)$

$R_1(A, B)$

A	B
a	b
d	b

$R_2(B, C)$

B	C
b	c
b	e

Szeretnénk visszaállítani a felbontás előtti relációt, így elvégezzük a természetes összekapcsolás műveletét a B közös attribútum segítségével (azokat a sorokat kapcsoljuk össze, melyeknél a közös attribútum értéka azonos:

$R_1 \bowtie R_2$

A	B	C
a	b	c
a	**b**	**e**
d	b	e
d	**b**	**c**

Az eredeti reláció 2 sora helyett 4 sort kaptunk, a vastag betűvel jelzettek az eredeti relációban nem szerepeltek. Ily módon **információt vesztettünk**, a **felbontás veszteséges.**

Miért mondjuk, hogy információt veszítettünk?
Több sorunk lett, mint eredetileg volt, ráadásul az eredeti reláció a dekomponálás után nem áll már rendelkezésünkre, így nem tudjuk leellenőrizni sem, hogy a természetes összekapcsolás műveletével kapott eredmény tényleg jó-e, a kiinduló adatokat kaptuk-e vissza.
Erre valami megoldást kell találni, mert a normalizálásra a redundáns adattárolás csökkentése, valamint a hatékonyság növelése miatt szükség van, de biztosítanunk kell, hogy a természetes összekapcsolás műveletének elvégzésével az eredeti relációt és a benne eredetileg szereplő adatokat kapjuk vissza.

Példa: Adott R(A, B, C, D, E) egy felbontása: R_1(A, B, C) és R_2(C, D, E)
E felbontás veszteséges, vagyis, az eredeti reláció nem állítható vissza természetes összekapcsolással, mert:
R egy előfordulása legyen r:

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b2	c1	d2	e2

ABC(r):

A	B	C
a1	b1	**c1**
a2	b2	**c1**

CDE(r):

C	D	E
c1	d1	e1
c1	d2	e2

ABC(r) $\bowtie$ **CDE(r)** (természetes összekapcsolás):

A	B	C	D	E
a1	b1	c1	d1	e1
a1	**b1**	**c1**	**d2**	**e2**
a2	**b2**	**c1**	**d1**	**e1**
a2	b2	c1	d2	e2

Nem lehet innen megállapítani melyek voltak az eredeti sorok: **információt vesztettünk**.

Veszteségmentesség

Példa:
Adott R(A, B, C, D, E), F={CDE} és R egy felbontása:
R1(A, B, C) és R2(C, D, E). R egy előfordulása legyen
R egy előfordulása legyen r:

A	B	C	D	E
a1	b1	c1	d1	e1
a2	b2	c1	?	??

ABC(r):

A	B	C
a1	b1	**c1**
a2	b2	**c1**

CDE(r):

C	D	E
c1	d1	e1
c1	?	??

ABC(r) $\bowtie$ **CDE(r)** (természetes összekapcsolás):

A	B	C	D	E
a1	b1	c1	d1	e1
a1	**b1**	**c1**	**?**	**??**
a2	**b2**	**c1**	**d1**	**e1**
a2	b2	c1	?	??

Az r akkor **legális**, ha ? helyén d1, ?? helyén e1 áll. Ekkor azonban CDE(r) egy sorra redukálódik: (c1, d1, e1). Igy nem keletkezhetnek "idegen sorok" a természetes összekapcsoláskor. Vagyis e felbontás **veszteségmentes.**
ENNEK OKA: C$\rightarrow$DE. C a közös attribútum!

Következmény: A felbontást a funkcionális függőségek alapján kell elvégezni. Az információvesztés 'idegen' sorok megjelenéséből áll. (Kevesebb sor sosem lesz!)

Lemma:

Az eredeti relációt a természetes összekapcsolásokkal kapott eredmény mindig tartalmazza:

$r = R_1(r) \bowtie R_2(r) \bowtie \ldots \bowtie R_k(r)$

Definíció: Az R reláció R_1, R_2, …R_k felbontása veszteségmentes az F függőségi halmaz szerint, ha

$r = R_1(r) \bowtie R_2(r) \bowtie \ldots \bowtie R_k(r)$ az R minden r előfordulására.

A Veszteségmentes felbontás a Boyce-Codd Normálformára bontás alapelve

Chase algoritmus (veszteségmentesség ellenőrzés)

Már adott felbontás veszteségmentességének eldöntésére szolgáló módszer.

Létre kell hoznunk egy táblázatot, melyben a sorok száma megegyezik a felbontásban szereplő relációk számával. Egy sor egy relációnak felel meg. Az oszlopok száma pedig az eredeti reláció attribútumainak számával egyezik meg.

Táblázat kitöltése:

i. sor k. eleme = a_k, ha a k. attribútum szerepel a sornak megfelelő reláció sémájában.

i. sor k. eleme = b_{ik}, ha a k. attribútum NEM szerepel a sornak megfelelő reláció sémájában.

Eljárás: Alkalmazzuk az F^+ függőségi halmaz elemeit :
Ha két sor megegyezik valamely oszlopban (legalább két a_k-t találunk), akkor a kiválasztott funkcionális függőség jobboldalának megfelelő oszlopban egyenlővé tesszük az értékeket. "a_i"-t írunk, ha valamelyik "a_i" volt, és "b_{ik}" - t, ha mindkettő "b_{jl}" volt. Ha „b"-ket írunk át, azt az összes előfordulásban kell, azokban a sorokban is, amelyekre a függés nem alkalmazható, de már az átírandó „b_{jl}" szerepel ott.

Döntés: Ha már nincsen alkalmazható függőség, akkor ellenőrizzük, az így kapott táblázatban van-e csupa "a_k"- ból álló sor. Ha igen, akkor a felbontás veszteségmentes. Ha nem, akkor veszteséges.

Példa:

Szállítók(Név, Cím, NYersanyag, Ár)
Szállítók(N, C, NY, Á)

A függőségek: N→C , N NY → Á
Vizsgáljuk meg, veszteségmentes-e az R_1(N, C) és R_2(N, NY, Á) felbontás?

A kiindulási táblázat a kitöltési szabály alkalmazásával:

	N	C	NY	Á
R_1	a_1	a_2	b_{13}	b_{14}
R_2	a_1	b_{22}	a_3	a_4

Alkalmazva az N→C függőséget:

	N	C	NY	Á
R_1	a_1	a_2	b_{13}	b_{14}
R_2	a_1	**a_2**	a_3	a_4

A második sorban **csupa "a_k"** áll, ezért a felbontás **veszteségmentes**.

Példa:

Adott R(A, B, C, D, E) és a függőségek halmaza:
F={A → B, B→D, E→A}.

R egy dekompozíciója R_1(A, B, C) és R_2(C, D, E).

Igazoljuk, hogy ez a dekompozíció veszteséges.

Igazolás: a fenti módszert alkalmazzuk

R	A	B	C	D	E
R_1	a_1	a_2	a_3	b_{14}	b_{15}
R_2	b_{21}	b_{22}	a_3	a_4	a_5

Mivel a sorok egyedül C-n egyeznek , csak olyan függőség jöhetne szóba, amelynek bal oldalán C áll. Mivel ilyen nincsen, a döntés adódik: a felbontás **veszteséges**.

Példa:

Adott:
R (A, B, C, D, E) és
F={A→C, B → C, C→D, D→E }

Az R (A, B, C, D, E) egy felbontása:
R_1(A, D), R_2(A, B), R_3(B, E), R_4(C, D, E), R_5(A, E)

Döntsük el, veszteségmentes-e ez a dekompozíció?

Megoldás:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{23}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{53}	b_{54}	a_5

A→C -t alkalmazva b_{13}, b_{23}, b_{53} egyenlővé válik b_{13}-mal

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	$\mathbf{b_{13}}$	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	$\mathbf{b_{13}}$	b_{54}	a_5

B→C -t alkalmazva b_{13}, b_{33} egyenlővé válik b_{13}-mal:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	**b_{13}**	b_{24}	b_{25}
R_3	b_{31}	a_2	**b_{13}**	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	b_{54}	a_5

C→D -t nem tudjuk alkalmazni, mert a C oszlopban nem találunk legalább két a_3-al jelölt sort:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{13}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	b_{54}	a_5

D→E -t alkalmazva b_{15} egyenlővé válik a_5-el:

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	**a_5**
R_2	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3	b_{31}	a_2	b_{13}	b_{34}	a_5
R_4	b_{41}	b_{42}	a_3	a_4	**a_5**
R_5	a_1	b_{52}	b_{13}	b_{54}	a_5

Több felhasználható függésünk nincs, így az Chase algoritmus véget ért.
Megnézzük találunk-e a táblázatban csupa a_k-ból álló sort. Sajnos nem, tehát ez a dekompozíció veszteséges.

Példa:

Adott:
R (A, B, C, D, E) és
F={A→C, B → C, C→D, D→E }

Az R (A, B, C, D, E) egy felbontása:
R_1(A, C), R_2(B, C), R_3(C, D), R_4(D, E)

Döntsük el, veszteségmentes-e ez a dekompozíció?

Megoldás:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}
R_2	b_{21}	a_2	a_3	b_{24}	b_{25}
R_3	b_{31}	b_{32}	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	b_{43}	a_4	a_5

A→C -t nem tudjuk alkalmazni, mert az A oszlopban nem találunk legalább két a_1-el jelölt sort

B→C -t nem tudjuk alkalmazni, mert az A oszlopban nem találunk legalább két a_2-el jelölt sort

C→D -t alkalmazva b_{13}, b_{33} egyenlővé válik a_4-el:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	**a_4**	b_{15}
R_2	b_{21}	a_2	a_3	**a_4**	b_{25}
R_3	b_{31}	b_{32}	a_3	**a_4**	b_{35}
R_4	b_{41}	b_{42}	b_{43}	a_4	a_5

D→E -t alkalmazva b_{15}, b_{25}, b_{35} egyenlővé válik a_5-el:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	**a_5**
R_2	b_{21}	a_2	a_3	a_4	**a_5**
R_3	b_{31}	b_{32}	a_3	a_4	**a_5**
R_4	b_{41}	b_{42}	b_{43}	a_4	**a_5**

Több felhasználható függésünk nincs, így az Chase algoritmus véget ért.
Megnézzük találunk-e a táblázatban csupa a_k-ból álló sort. Sajnos nem, tehát ez a dekompozíció veszteséges.
Miért?
Már figyeltünk arra, amit korábban tanultunk, hogy a függőségek mentén végeztük el a dekomponálást!

A válasz az, hogy nincs a felbontott relációk között sorra elvégezve a természetes összekapcsolás műveletét, nem kapjuk vissza az eredeti relációt. Szükségünk van egy un. kapcsolótáblára, mely legalább a kulcs attribútumokat tartalmazza.

Példa:
Határozzuk meg a R (A, B, C, D, E) reláció kulcsát az érvényes funkcionális függések
F={A→C, B → C, C→D, D→E }
alapján az attribútumok lezártjai segítségével:

$\{A\}^+=\{A, C, D, E\}$
$\{B\}^+=\{B, C, D, E\}$
$\{C\}^+=\{C, D, E\}$
$\{D\}^+=\{D, E\}$
$\{E\}^+=\{E\}$

$\{A,B\}^+ =\{A, B, C, D, E\}$
$\{A,C\}^+ =\{A, C, D, E\}$
$\{A,D\}^+ =\{A, D, C, E\}$
$\{B,C\}^+ =\{B, C, D, E\}$
$\{B,D\}^+ =\{B, C, D, E\}$
$\{C,D\}^+ =\{C, D, E\}$

A kulcs az AB attribútumhalmaz
Végezzük el az R (A, B, C, D, E) dekomponálását, a függőségek mentén újra, figyelve arra, hogy legyen olyan relációnk, mely legalább a kulcs attribútumokat tartalmazza.

függőségek szerinti felbontás:

$R_1(A, C)$, $R_2(B, C)$, $R_3(C, D)$, $R_4(D, E)$

nincs olyan relációnk, mely legalább az AB attribútumokat tartalmazza, így külön létre kell hoznunk egyet:

$R_5(A, B)$

Döntsük el, veszteségmentes-e ez a dekompozíció?

Megoldás:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}
R_2	b_{21}	a_2	a_3	b_{24}	b_{25}
R_3	b_{31}	b_{32}	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	b_{43}	a_4	a_5
R_5	a_1	a_2	b_{53}	b_{54}	b_{55}

$A \rightarrow C$ -t alkalmazva b_{53} egyenlővé válik a_3-al:

	A	B	C	D	E
R_1	a_1	b_{12}	$\mathbf{a_3}$	b_{14}	b_{15}
R_2	b_{21}	a_2	a_3	b_{24}	b_{25}
R_3	b_{31}	b_{32}	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	b_{43}	a_4	a_5
R_5	a_1	a_2	$\mathbf{a_3}$	b_{54}	b_{55}

$\underline{B} \rightarrow \underline{C}$ -t alkalmazva nem történik változás, hiszen már minden a_2-t tartalmazó sorban a_3 szerepel a C oszlopban:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}
R_2	b_{21}	$\mathbf{a_2}$	$\mathbf{a_3}$	b_{24}	b_{25}
R_3	b_{31}	b_{32}	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	b_{43}	a_4	a_5
R_5	a_1	$\mathbf{a_2}$	$\mathbf{a_3}$	b_{54}	b_{55}

$C \rightarrow D$ -t alkalmazva b_{14}, b_{24}, b_{54}, egyenlővé válik a_4-el:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	$\mathbf{a_4}$	b_{15}
R_2	b_{21}	a_2	a_3	$\mathbf{a_4}$	b_{25}
R_3	b_{31}	b_{32}	a_3	$\mathbf{a_4}$	b_{35}
R_4	b_{41}	b_{42}	b_{43}	a_4	a_5
R_5	a_1	a_2	a_3	$\mathbf{a_4}$	b_{55}

$D \rightarrow E$ -t alkalmazva b_{15}, b_{25}, b_{35}, és b_{54}, egyenlővé válik a_5-el:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	$\mathbf{a_5}$
R_2	b_{21}	a_2	a_3	a_4	$\mathbf{a_5}$
R_3	b_{31}	b_{32}	a_3	a_4	$\mathbf{a_5}$
R_4	b_{41}	b_{42}	b_{43}	a_4	$\mathbf{a_5}$
R_5	a_1	a_2	a_3	a_4	$\mathbf{a_5}$

Ha megnézzük a táblázatot, akkor azt látjuk, hogy az R_5-ös sorban csupa a_k-ból álló érték szerepel, tehát **a felbontás veszteségmentes.**

Vizsgáljuk meg, hogy a felbontásunk függőségőrző-e?

A dekomponálásnál az alábbi relációkat kaptuk:
$R_1(A, C)$, $R_2(B, C)$, $R_3(C, D)$, $R_4(D, E)$, $R_4(A, B)$

Az egyes relációkhoz tartozó funkcionális függőségi halmazok:

$F_1=\{A \rightarrow C\}$
$F_2=\{B \rightarrow C\}$
$F_3=\{C \rightarrow D\}$
$F_4=\{D \rightarrow E\}$
$F_5=\{\}$

Fontos megjegyzés:

Az egyik legfontosabb normálforma az ún. Boyce-Codd normálforma, röviden BCNF (ld. következő előadás). Ezért fontos tudni, hogy **mindig van veszteségmentes** felbontás BCNF-re.

Függőségőrző felbontás

A veszteségmentesség mellett a felbontás másik fontos kritériuma, hogy a felbontás után kapott relációkban meglevő függőségekből az eredeti függőségekre tudunk-e következtetni. Ha nem, akkor a függőségek ellenőrzése csak a természetes összekapcsolással lehetséges, ami nagyon költséges. Ezért fontos, ha lehet, a függőségmegőrző dekompozíció megvalósítása.

Példa:
Adottak: R(A, B, C)
F={A→B, B→C}

R egy felbontása: R_1(A, C) R_2(B, C)
Vizsgáljuk meg, függőségőrző-e ez a felbontás?

Megoldás:
R_1-beli ún. **vetített függőségek**:

F_1={A→A, C→C, A→C}

Megjegyzés: A→C is vetített függőség, F^+-ból kell vetíteni, hiszen nyilván az eredeti A→B, B→C függőségekből A→C azonnal adódik a tranzitivitási szabállyal. Tehát A→C-nek minden reláció előfordulás eleget tesz, így a dekomponálás után létrejövő reláció előfordulások is eleget tesznek.

R_2-beli **vetített függőségek:** F_2= {B→B, C→C, B→C}

A nem triviális függőségek F_1-ben és F_2-ben B→C és A→C. Ezekből A→B nem vezethető le, A→B funkcionális függőség ”elveszett” tehát a felbontás **nem függőségőrző.**

Adottak: R(A, B, C)
F={A→B, B→C}

Ha R-et az R_1’=(A, B) és R_2’=(B, C) relációkra bontottuk volna fel, akkor a vetített függőségek:

F_1’={A→B, A→A, B→B}

F_2’={B→B, C→C, B→C}

F’= F_1’ U F_2’ = {A→B, A→A, B→B, B→B, C→C, B→C}

$(F')^+=F^+$, hiszen az eredetileg adott F elemei szerepelnek F’-ben (így ugyanazok a függőségek vezethetők le.)

Veszteségmentes felbontás: az eredeti reláció-előfordulás visszakapható a felbontásban szereplő relációkból. A felbontásban az előfordulások R vetítettjei.

Függőségőrző felbontás (informális): Adott az R reláció és az F függőségi halmaz. Azt mondjuk, hogy az R reláció egy R_1, R_2, …R_k felbontása **függőségőrző** az F függőségi halmaz szerint, ha az R_1, R_2, …R_k relációkra **vetített függőségek** unióiból az eredeti F függőségi halmaz logikailag következik. (Pl. levezethető az Armstrong axiómákkal)

Függőségek vetítése: Az F függőségi halmaznak a Z attribútumhalmazra való F_z vetítettje azokat a függőségeket tartalmazza, amelyek benne vannak F^+-ban, és a bennük szereplő attribútumok szerepelnek Z-ben:

$$\Pi_z(F^+)=\{ X\rightarrow Y \mid \{X,Y\}\subseteq Z \text{ és } X\rightarrow Y \in F^+ \}$$

Példa:
R(A, B, C, D, E), F={ $A \rightarrow D$, $B \rightarrow E$, $DE \rightarrow C$ }.
S(A, B, C) tagja az R egy felbontásának. Mely függőségek érvényesek S-ben?

Megoldás: Ki kell számolni az {A, B, C} halmaz minden részhalmazának lezártját:
$\{A\}^+ = AD$, $A \rightarrow D$, nincs új függés
$\{B\}^+ = BE$, de E nincsen a sémában, így $B \rightarrow E$ nem érvényesül
$\{C\}^+ = C$, nincs új függés
$\{AB\}^+ = ABCDE$, így $AB \rightarrow C$ érvényes S-ben (DE nincsen S-ben)
$\{BC\}^+ = BCE$, nincs új függés
$\{AC\}^+ = ACD$, nincs új függés
$\{ABC\}^+ = ABCDE$, nincs új függés

Csak az $AB \rightarrow C$ nemtriviális új függőséget kaptuk, igy $\Pi_S(F^+)=\{A\rightarrow C, +$ triviálisak$\}$

Az F függőségi halmaznak valamely relációra való vetítettjén a reláció sémájában szereplő attribútumhalmazra való vetítettjét értjük.

Függőségőrzés definíciója (formálisabb): Legyen az R egy felbontása R_1, R_2, …R_k. Az ezekre vetített megfelelő függőségi halmazok rendre F_1, F_2, …F_k. Azt mondjuk, hogy e felbontás függőségőrző, ha

$$\{F_1 \cup F_2 \cup \ldots \cup F_k\}^+=F^+$$

$$F_i=\Pi_{Ri}(F^+) =\{ X\rightarrow Y \mid \{X,Y\}\subseteq R_i \text{ és } X\rightarrow Y \in F^+ \}$$

Definíció: Két függőségi halmaz, G és F ekvivalens, ha G logikai következménye F, és F logikai következménye G, vagyis $G^+=F^+$.

Ez azt jelenti, hogy akkor függőségőrző egy felbontás, ha $\{F_1 \cup F_2 \cup \ldots \cup F_k\}$ és F ekvivalensek.

Jelentősége: minden függőség az adott relációra előírt kényszerfeltételnek fogható fel. E feltételeket minden frissítéskor (update), beszúráskor(insert), törléskor (delete, drop) ellenőrizni kell. Ha a vetített függőségekből az eredeti nem következik, akkor a feltételek teljesülését a relációk közötti ellenőrzéssel, természetes összekapcsolásokkal lehet csak ellenőrizni. Ha a függőségek is megőrződnek, akkor viszont elegendő a dekomponált relációkra előírt kényszerfeltételek betart(tat)ása.

A függőségmegőrzés ellenőrzésének komplexitása exponenciális.

Példa:

R(VÁROS, UTCA, IRÁNYÍTÓSZÁM)=R(V, U, Ir)
F={VU→Ir, Ir→V}

Határozzuk meg a reláció kulcsait és azt, hogy hányadik normálformában van!
R felbontása BCNF-re: R_1(U, Ir) és R_2(V, Ir) veszteségmentes felbontás, hiszen:

	V	U	Ir
R_1	b_{11}	a_2	a_3
R_2	a_1	b_{22}	a_3

Alkalmazva az Ir→V függőséget, b_{11} és a_1 egyenlő lesz, a_1-gyel:

	V	U	Ir
R_1	a_1	a_2	a_3
R_2	a_1	b_{22}	a_3

Az első sorból következtetünk a **veszteségmentesség**re.

Azonban a függőségek nem őrződnek meg, hiszen
F={VU→Ir, Ir→V} vetítettjei:
R_1(U, Ir)-re, F_1={U→U, Ir→Ir}
R_2(V, Ir)-re, F_2={ Ir→V, Ir→Ir , V→ V}.

Az eredeti F azonban nem vezethető le, mert:
$F_1 \cup F_2$={U→U, Ir→Ir, Ir→V, V→ V }.
VU→ Ir akkor és csak akkor vezethető le, ha Ir$\in${V,U}$^+$.
{V,U}$^+$={V, U} , vagyis Ir nincsen benne a lezártban, ezért a VU→ Ir függőség nem következik a vetített függőségi halmazokból.

Az előző példa azt illusztrálta, hogy a veszteségmentes felbontás BCNF-re nem biztos, hogy függőségőrző. Most azt ellenőrizzük, hogy e példabeli esetben nem is létezik függőségőrző felbontás BCNF-re.

Példa:

R(VÁROS, UTCA, IRÁNYÍTÓSZÁM)=R(V, U, Ir)
F={VU→Ir, Ir→V}

Kulcsjelöltek: {V, U} és {U, Ir}

A reláció 3NF-ben van.
Láttuk, hogy az R1(U, Ir), R2(V, Ir) felbontás BCNF-re veszteségmentes, de nem függőségőrző

Feladat: Ellenőrizzük, hogy a (V, U), (U, Ir) felbontás, és az (V, Ir) és (U, V) sem függőségőrző!

Következmény: Nem mindig létezik függőségmegőrző felbontás BCNF-re

Megjegyzés: Mindig létezik veszteségmentes és függőségmegőrző felbontás 3NF-re.

Feladat a vetített függőségek meghatározására:

R (A, B, C, D) felbontása: R_1(A, B, C),
R_2(C, D)

F={B→C, AC→D}
Az alábbi F szerinti lezártak segítségével döntsük el, mely függőségek érvényesek R_1-ben és R_2-ben!

(**Útmutatás**: Ha α^+ adott, és $\beta \in \alpha^+$, akkor minden $\alpha \rightarrow \beta$függőség levezethető, pl. alább
$\{B,D\}^+ = \{B, C, D\}$
összefüggésből látható, hogy pl. BD→C fennáll.)

$\{A\}^+ = \{A\}$
$\{B\}^+ = \{B, C\}$
$\{C\}^+ = \{C\}$
$\{A,B\}^+ = \{A, B, C, D\}$
$\{A,C\}^+ = \{A, C, D\}$
$\{A,D\}^+ = \{A, D\}$
$\{B,C\}^+ = \{B, C\}$
$\{B,D\}^+ = \{B, C, D\}$
$\{C,D\}^+ = \{C, D\}$
$\{A,B,C\}^+ = \{A, B, C, D\}$
$\{A,B,D\}^+ = \{A, B, C, D\}$
$\{B,C,D\}^+ = \{B, C, D\}$
$\{A,C,D\}^+ = \{A, C, D\}$
$\{A,B, C,D\}^+ = \{A, B, C, D\}$

Példa:

R(A, B, C)
F={A→B, B→C }

Az alábbi felbontás veszteségmentes és függőségőrző:
R_1(A,B)
R_2(B,C)
$R_1 \cap R_2$=B, és B→C teljesül, ezért veszteségmentes.
Mindkét adott függőség benne van a $F_1 \cup F_2$ halmazban, ezért veszteségmentes.

Az alábbi felbontás veszteségmentes de nem függőségőrző:

R_1(A,B)
R_2(A,C)

F_1={ A→B, triviálisak } F_2={ A→C, triviálisak}
B→C nem vezethető le az $F_1 \cup F_2$ vetített függőségi halmazok uniójából, hiszen erre vonatkoztatva $B^+=\{B\}$.

Példa: Legyen R(A,B,C), és F:={AB→C, C→B}
Igazoljuk, hogy R-nek nem létezik veszteségmentes, függőségőrző dekomponálása BCNF-re!

Megoldás:
R kulcsa AB és AC, ezért nincsen BCNF-ben, C→B baloldala nem szuperkulcs. Az ismert módszer szerint dekomponálva:
R_1(C,B)
R_2(A,C)

A dekompozíció veszteségmentes, hiszen C→B fennáll.
A dekompozíció nem függőségőrző, mert AB→C nem vezethető le.

Más dekompozíciókkal próbálkozva:
R_1(A,B)
R_2(B,C)
Ez veszteségmentes, de nem függőségőrző, mert AB→C nem vezethető le.

Egyetlen további lehetőség van:
R_1(A,B)
R_2(A,C)
Ez a dekompozíció pedig még csak nem is veszteségmentes.

Megjegyzés: Az eredeti R reláció 3NF-ben van, mert B elsődleges attribútum. Tehát, ha megelégszünk a 3NF-fel, akkor nem is kell felbontani. Gyakran ez a célszerűbb, a függőségek megőrzése fontosabb lehet, mint a még fennálló redundancia. Ezt a konkrét feladat dönti el.
Feladat: R(Bérlő_az, Vitorlás_az, Dátum), röviden R(B, V, D).

A klub egy napon csak egy hajót ad kölcsön, és egy személy egy hajót egy napra bérelhet. Ezek szerint a következő funkcionális függőségek írhatók elő: BV→D, D→V. A kulcs megállapítása után döntse el, hányadik normálformában van a reláció. Van-e veszteségmentes, függőségőrző felbontás BCNF-re?

Veszteségmentes, függőségőrző felbontás 3NF-re

Definíció: F bázisa az a függőségi halmaz, amelyből F függőségei levezethetők. **Minimális** e bázis, ha nincsen olyan valódi részhalmaza, amely bázis lenne.

A minimális bázis azt jelenti, hogy nem lehet egyetlen függőséget sem elhagyni, mert e részhalmaz már nem lesz ekvivalens F-fel.

Definíció: F minimális fedése, F^-, **olyan minimális bázis**, amelyben szereplő függőségek bal és jobboldala minimális a következő értelemben:

1. Minden függőség jobb oldala egyetlen attribútum (felbontási szabály)

2. Nem lehet egyetlen $X \rightarrow Y$ függőséget sem helyettesíteni olyan $W \rightarrow Y$ függőséggel, hogy $W \subset X$ és a kapott halmaz ekvivalens F-fel

Összefoglalva:

Bázis: Lezártja tartalmazza F összes következményét

Minimális fedés: mindegyik függőség minimális, vagyis függőségben minden attribútum szükséges a bal oldalon és egyetlen attribútum áll a jobb oldalon, és nincsen a bázisnak olyan valódi részhalmaza, amelynek lezártja F összes következményét tartalmazná.

Következmény: A minimális fedésben $\alpha \rightarrow Y$ alakú függőségek lesznek
Lemma: minden függőségi halmaznak van minimális bázisa (ez nem egyértelmű, több is lehet)

Példa:

$F=\{A \rightarrow B, ABCD \rightarrow E, EF \rightarrow G, EF \rightarrow H, ACDF \rightarrow EG\}$
Határozzuk meg e funkcionális függőségi rendszer minimális bázisát!

Megoldás:
1. $ACDF \rightarrow EG$ függőségből szétválasztással: $ACDF \rightarrow E$ és $ACDF \rightarrow G$ adódik.
2. *

$ACDF \rightarrow G$ levezethető az $A \rightarrow B$, $ABCD \rightarrow E$ és $EF \rightarrow G$ függőségekből:
$A \rightarrow B$ miatt $ABCD \rightarrow E$ –ben B helyére A-t írunk:
$ACD \rightarrow E$ (pszeudotranzitivitás) majd F-fel bővítve:
$ACDF \rightarrow EF$, végül tranzitivitással $EF \rightarrow G$ segítségével: $ACDF \rightarrow G$. Ezért $ACDF \rightarrow G$ törölhető.
*
Hasonlóan $ACDF \rightarrow E$ is levezethető: $A \rightarrow B$ miatt $ABCD \rightarrow E$ –ben kapjuk $ACD \rightarrow E$-t, F-fel bővítve: $ACDF \rightarrow EF$, szétválasztva: $ACDF \rightarrow E$ (és $ACDF \rightarrow F$)
Igy $ACDF \rightarrow E$ is törölhető.

3. Fentiek alapján tudjuk azt is, hogy ABCD→E-ben B felesleges a bal oldalon, helyettesíthető A-val, így a maradék A→B, ACD→E, EF→G, EF→H funkcionális függőségek baloldala minimális, egyik attribútum sem hagyható el. Ugyanis abból, hogy ACD EGYÜTTESEN meghatározza E-t, nem következik, hogy kettő a bal oldalról meghatározza E-t (fordítva igaz, a baloldalt szabad "büntetlenül" bővíteni). Ha ez igaz lenne, akkor a fennálló maradék rendszerből le lehetne vezetni.

Pl. ha ACD-ből elhagynánk D-t, akkor
A→B, ACD→E, EF→G, EF→H függőségi halmaz szerinti lezárt:
AC^+={A, C, B}, tehát AC→E nem vezethető le.
Hasonlóan belátható, hogy sem A, sem C nem hagyható el:
CD^+={C, D}, CD →E tehát nem vezethető le,
AD^+={A, D, B}, CD →E tehát nem vezethető le.

EF →G esetében: E^+ ={E}, F^+={F}, és hasonlóan EF→H-ra is.

4. Azt kellene még egyszer ellenőrizni, hogy nincsen-e felesleges függőség, ami elhagyható. Mivel a jobb és bal oldalakon szereplő attribútumok között csak az E a közös (bal: ACDEF, jobb: EGHB), ezért csak az EF→H és EF→G szerepe lehet kétséges. Akármelyiket hagyjuk is el, a maradék rendszerből nem lehetne őket sem levezetni, így F^+-t sem, pl. kimaradna továbbá ACDF →H ha EF →H-t törölnénk.

Veszteségmentes, függőségőrző felbontás 3NF-re I.

1. Keressük meg F minimális fedését, legyen ez F^-.
2. Keressünk meg az F- halmazban minden olyan függőséget, amelynek baloldala
α: $\alpha \rightarrow Y_1$, $\alpha \rightarrow Y_2$,…. $\alpha \rightarrow Y_k$, és minden ilyenre készítsük el az (α,Y_1, Y_2, …Y_k sémát) (függőségőrzés)
3. A maradék attribútumokkal képezzünk egy másik sémát (veszteségmentesség – minden attribútumnak szerepelnie kell)
4. Ha egyik sem tartalmaz egyetlen kulcsjelöltet sem, akkor egy kulcsjelölttel is készítsünk egy sémát. (veszteségmentesség, joinnal az eredeti visszállítható legyen)

Veszteségmentes, függőségőrző felbontás 3NF-re II.

1. F^- alapján megkíséreljük a BCNF-re hozást, vagyis a sértő függőségeket elsősorban az F^- halmazban keressük.
2. Ha nem sikerül minden függőséget az F^- halmazból megőrizni, a kimaradó függőségekkel létrehozzuk a szükséges relációkat.

Veszteségmentes, függőségőrző felbontás 3NF-re III:

1. A szokásos módon létrehozzuk a veszteségmentes BCNF dekompozíciót.
2. Ellenőrizzük, minden F^- halmazbeli függőség megőrződött-e. Ha nem, a kimaradókkal újabb relációkat képezünk.

Veszteségmentes, függőségőrző felbontás BCNF-re

Az esetek többségében tudunk adni veszteségmentes és függőségőrző felbontást BCNF-re, de a tranzitív függések esetében jobban kell figyelnünk.

Példa:

Adott:
R (A, B, C, D, E) és
F={A→C, B→D, C→E}

Készítsük el a reláció veszteségmentes és függőségőrző felbontását BCNF-re.

Első lépés a kulcs meghatározása az attribútum lezártak segítségével:
A^+={A, C, E} a tranzitív C→E függés miatt
B^+={B, D}
C^+={C, E}
D^+={D}
E^+={E}
Az egy attribútumból álló lezártak közül egyet sem találtunk, melyben a reláció összes mezője szerepelne, ezért folytatjuk a két attribútumból álló lezártak kifejtésével.

AB^+={A, B, C, D, E}
AC^+={A, C, E}
AD^+={A, D, C, E}
AE^+={A, E, C}
BC^+={B, C, D, E}
BD^+={B, D}
BE^+={B, E, D}
CD^+={C, D, E}
CE^+={C, E}
DE^+={D, E}

Kulcsjelölt és kulcs az AB attribútumhalmaz, mert az ő lezártjában szerepel a reláció összes attribútuma.

A függőségek és a kulcs (kapcsolótábla) figyelembevételével az R (A, B, C, D, E) egy BCNF felbontása:
R_1(A, C), R_2(B, D), R_3(C, E), R_4(A, B)
F_1={A→C}
F_2={B→D}
F_3={C→E}
F_4={}

Döntsük el, veszteségmentes-e ez a dekompozíció?

Megoldás:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}
R_3	b_{31}	b_{32}	a_3	b_{34}	a_5
R_4	a_1	a_2	b_{43}	b_{44}	b_{45}

$A \rightarrow C$-t alkalmazva b_{43} egyenlővé válik a_3-al:

	A	B	C	D	E
R_1	a_1	b_{12}	$\mathbf{a_3}$	b_{14}	b_{15}
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}
R_3	b_{31}	b_{32}	a_3	b_{34}	a_5
R_4	a_1	a_2	$\mathbf{a_3}$	b_{44}	b_{45}

$B \rightarrow D$-t alkalmazva b_{44} egyenlővé válik a_4-el:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}
R_2	b_{21}	a_2	b_{23}	$\mathbf{a_4}$	b_{25}
R_3	b_{31}	b_{32}	a_3	b_{34}	a_5
R_4	a_1	a_2	a_3	$\mathbf{a_4}$	b_{45}

$C \rightarrow E$-t alkalmazva b_{15} és b_{45} egyenlővé válik a_5-el:

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	b_{14}	$\mathbf{a_5}$
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}
R_3	b_{31}	b_{32}	a_3	b_{34}	a_5
R_4	a_1	a_2	a_3	a_4	$\mathbf{a_5}$

Ha megnézzük a táblázatot, akkor azt látjuk, hogy az R_4-es sorban csupa a_k-ból álló érték szerepel, tehát **a felbontás veszteségmentes.**

Vizsgáljuk meg, hogy a felbontásunk függőségőrző-e?

A felbontásnál az alábbi relációkat és hozzájuk tartozó függőségi halmazokat kaptuk:
$R_1(A, C)$, $R_2(B, D)$, $R_3(C, E)$, $R_4(A, B)$
$F_1=\{A \rightarrow C\}$
$F_2=\{B \rightarrow D\}$
$F_3=\{C \rightarrow E\}$
$F_4=\{\}$

Az a kérdés, hogy a felbontott relációk függőségi halmazainak uniója megegyezik-e az eredeti reláció függőségi halmazával. Ha igen, a felbontás függőségőrző, ha nem, akkor a felbontás nem függőségőrző.

$F_1 \cup F_2 \cup F_3 \cup F_4 \equiv F$?

Az eredeti F={A→C,B →D, C→E} függőségi halmaz minden elemét visszakapjuk, ha az F_1 U F_2 U F_3 U F_4-et elvégezzük, tehát a felbontás **Függőségőrző**.

Példa:
Az előző példában átugrottuk a BCNF-re hozás lépéseit, ezért nézzünk most egy teljesen kifejtett példát.

Adott:
R (A, B, C, D, E, F) és
F={A→C, B→D, C→E, D→F}

Készítsük el a reláció veszteségmentes és függőségőrző felbontását BCNF-re.

Első lépés a kulcs meghatározása az attribútum lezártak segítségével:
A^+={A, C, E} a tranzitív C→E függés miatt
B^+={B, D, F} a tranzitív D→F függés miatt
C^+={C, E}
D^+={D, F}
E^+={E}
F^+={F}
Az egy attribútumból álló lezártak közül egyet sem találtunk, melyben a reláció összes mezője szerepelne, ezért folytatjuk a két attribútumból álló lezártak kifejtésével.
AB^+={A, B, C, D, E, F}
AC^+={A, C, E}
AD^+={A, D, C, E, F}
AE^+={A, E, C}
AF^+={A, F, C, E}
BC^+={B, C, D, E, F}
BD^+={B, D, F}
BE^+={B, E, D, F}
BF^+={B, F, D}
CD^+={C, D, E, F}
CE^+={C, E}
CF^+={C, F, E}
DE^+={D, E, F}
DF^+={D, F}
EF^+={E, F}

A két attribútumból álló lezártak közül az AB^+, amely az összes attribútumot tartalmazza a relációból, ezért AB a kulcs.

Kezdjük el a BCNF-re hozást:
R (A, B, C, D, E, F) és
F={A→C, B→D, C→E, D→F}

Az első függőség, ami sérti a BCNF-et az A→C, tehát eszerint végezzük a dekomponálást. A függéshez tartozó attribútumok kerülnek az egyik relációba pl.R_1-be, a függés bal oldalán lévő attribútum és a reláció többi attribútuma kerül a másik relációba. Ezt a műveletsort végezzük addig, amíg a BCNF-et sértő függések el nem fogynak.

$R_1(A, C)$; $F_1=\{A\rightarrow C\}$

$R_2(A, B, D, E, F)$; $F_2=\{B\rightarrow D, D\rightarrow F\}$

a $C\rightarrow E$ funkcionális függés nem érvényes R_1-ben sem R_2-ben, mert ahhoz mindkét attribútumnak (A-nak és C-nek) benne kellene lennie az adott relációban, de ez nem teljesül.

R_1 kulcsa **A** a függőségi halmazból lezártak segítségével levezetve:
$A^+=\{A, C\}$
$C^+=\{C\}$

Mivel R_1-ben a funkcionális függőségi halmazban csak egy nem triviális függés van és ennek a bal oldala szuperkulcs (legalább a kulcs attribútumo(ka)t tartalmazza), ezért BCNF-ben van.

R_2 kulcsa **ABE** a függőségi halmazból lezártak segítségével levezetve (Házi feladat).

Mivel mindkét funkcionális függőség baloldala nem szuperkulcs, így R_2 nincs BCNF-ben.
Folytatjuk tehát a BCNF-re hozást a korábbi módszer alkalmazásával. pl. $B\rightarrow D$ függés mentén:

$R_{21}(B, D)$; $F_{21}=\{B\rightarrow D\}$

$R_{22}(A, B, E, F)$; $F_{22}=\{\}$;
a $B\rightarrow D$, $D\rightarrow F$ funkcionális függés nem érvényes R_{22}-ben, ezért csak triviális függéseket tartalmaz.

A R_{21} kulcsa **B** a függőségi halmazból lezártak segítségével levezetve:
$B^+=\{B, D\}$
$D^+=\{D\}$

Mivel R_{21}-ben a funkcionális függőségi halmazban csak egy nem triviális függés van és ennek a bal oldala szuperkulcs (legalább a kulcs attribútumo(ka)t tartalmazza), ezért BCNF-ben van.

$R_{22}(A, B, E, F)$; $F_{22}=\{\}$

Az R_{22} függőségi halmaza csak triviális függéseket tartalmaz, ezért nem sérti a BCNF kritériumát, tehát BCNF-ben van.

A kiinduló R (<u>A</u>, <u>B</u>, C, D, E, F) relációt BCNF-re bontottuk az alábbiak szerint:

$R_1(A, C)$; $F_1=\{A\rightarrow C\}$
$R_{21}(B, D)$; $F_{21}=\{B\rightarrow D\}$
$R_{22}(A, B, E, F)$; $F_{22}=\{\}$

Veszteségmentes a felbontás?

A veszteségmentességet az F={A→C, B→D, C→E, D→F} funkcionális függőség halmaz alapján az ismert Chase algoritmussal ellenőrizhetjük:

	A	B	C	D	E	F
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}	b_{16}
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}	b_{26}
R_3	a_1	a_2	b_{33}	b_{34}	a_5	a_6

A→C-t alkalmazva b_{33} egyenlővé válik a_3-al:

	A	B	C	D	E	F
R_1	a_1	b_{12}	$\mathbf{a_3}$	b_{14}	b_{15}	b_{16}
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}	b_{26}
R_3	a_1	a_2	$\mathbf{a_3}$	b_{34}	a_5	a_6

B→D-t alkalmazva b_{34} egyenlővé válik a_4-el:

	A	B	C	D	E	F
R_1	a_1	b_{12}	a_3	b_{14}	b_{15}	b_{16}
R_2	b_{21}	a_2	b_{23}	$\mathbf{a_4}$	b_{25}	b_{26}
R_3	a_1	a_2	a_3	$\mathbf{a_4}$	a_5	a_6

C→E-t alkalmazva b_{15} egyenlővé válik a_5-el:

	A	B	C	D	E	F
R_1	a_1	b_{12}	a_3	b_{14}	$\mathbf{a_5}$	b_{16}
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}	b_{26}
R_3	a_1	a_2	a_3	a_4	$\mathbf{a_5}$	a_6

D→F-t alkalmazva b_{26} egyenlővé válik a_6-al:

	A	B	C	D	E	F
R_1	a_1	b_{12}	a_3	b_{14}	a_5	$\mathbf{a_6}$
R_2	b_{21}	a_2	b_{23}	a_4	b_{25}	b_{26}
R_3	a_1	a_2	a_3	a_4	a_5	$\mathbf{a_6}$

Ha megnézzük a táblázatot, akkor azt látjuk, hogy az R_3-as sorban csupa a_k-ból álló érték szerepel, tehát **a felbontás veszteségmentes.**

Nézzük meg, hogy az eredeti funkcionális függőségi halmaz alapján F={A→C, B→D, C→E, D→F} függőségőrző lett-e a felbontás?

F_1={A→C}
F_{21}={B→D}
F_{22}={}

$F_1 \cup F_{21} \cup F_{22} \equiv F$?
$F_1 \cup F_{21} \cup F_{22}$ ={A→C, B→D} ≠ F

Elveszítettük a felbontás során a C→E, D→F függőségeket, tehát a felbontás **nem függőségőrző!**

Mi történhetett?
A tranzitív függéseknél már említettük, hogy a második tag alapján kell a dekomponálást megkezdeni, de mi mindkét tranzitív függés esetében az első tag szerint bontottuk fel az adott relációt, ez vezetett a függés második tagjának elvesztéséhez.

Most kezdjük újra a feladatot, de már figyelve a tranzitív függéseknél a dekomponálásra.

R (<u>A</u>, <u>B</u>, C, D, E, F) és
F={A→C, B→D, C→E, D→F}

Megnézzük melyek a tranzitív függések: A→C, C→E és a B→D, D→F
Mindegyik függés sérti jelenleg a BCNF feltételét, mivel AB a kulcs, így az egyik tranzitív függés második tagjával kezdjük a felbontást.

Az első tranzitív függés második tagja, ami sérti a BCNF-et az C→E, tehát eszerint végezzük a dekomponálást. A függéshez tartozó attribútumok kerülnek az egyik relációba pl. R_1-be, a függés bal oldalán lévő attribútum és a reláció többi attribútuma kerül a másik relációba. Ezt a műveletsort végezzük addig, amíg a BCNF-et sértő függések el nem fogynak.

R_1(C, E); F_1={C→E}

R_2(A, B, C, D, F); F_2={A→C, B→D, D→F}

R_1 kulcsa **C** a függőségi halmazból lezártak segítségével levezetve:
C^+={C, E}
E^+={E}

Mivel R_1-ben a funkcionális függőségi halmazban csak egy nem triviális függés van és ennek a bal oldala szuperkulcs (legalább a kulcs attribútumo(ka)t tartalmazza), ezért BCNF-ben van.

R_2 kulcsa **ABC** a függőségi halmazból lezártak segítségével levezetve (házi feladat).

$R_2(A, B, C, D, F)$; $F_2=\{A\rightarrow C, B\rightarrow D, D\rightarrow F\}$ esetében szintén találunk egy BCNF-et sértő tranzitív függést, így ismét a második tagja $D\rightarrow F$ szerint kezdjük a felbontást.

$R_{21}(D, F)$; $F_{21}=\{D\rightarrow F\}$

$R_{22}(A, B, C, D)$; $F_{22}=\{A\rightarrow C, B\rightarrow D\}$

R_{21} kulcsa **D** a függőségi halmazból lezártak segítségével levezetve:
$D^+=\{D, F\}$
$F^+=\{F\}$

Mivel R_{21}-ben a funkcionális függőségi halmazban csak egy nem triviális függés van és ennek a bal oldala szuperkulcs (legalább a kulcs attribútumo(ka)t tartalmazza), ezért BCNF-ben van.

R_{22} kulcsa **AB** a függőségi halmazból lezártak segítségével levezetve:
$A^+=\{A, C\}$
$B^+=\{B, D\}$
$C^+=\{C\}$
$D^+=\{D\}$

Az egy attribútumból álló lezártak közül egyet sem találtunk, melyben a reláció összes mezője szerepelne, ezért folytatjuk a két attribútumból álló lezártak kifejtésével.

$AB^+=\{A, B, C, D\}$
$AC^+=\{A, C\}$
$AD^+=\{A, D, C\}$
$BC^+=\{B, C, D\}$
$BD^+=\{B, D\}$
$CD^+=\{C, D\}$

A két attribútumból álló lezártak közül az AB^+, amely az összes attribútumot tartalmazza a relációból, ezért AB a kulcs.

Az $R_{22}(A, B, C, D)$; $F_{22}=\{A\rightarrow C , B\rightarrow D\}$ reláció esetében már nem találunk tranzitív függést, de mindkét nem triviális függés sérti a BCNF definícióját, most már szabadon választhatunk a két függés közül, melyik mentén folytatjuk a dekomponálást.

Válasszuk pl. a $B\rightarrow D$ függést:

$R_{221}(B, D)$; $F_{221}=\{B\rightarrow D\}$

$R_{222}(A, B, C)$; $F_{222}=\{A\rightarrow C\}$

Az R_{221} kulcsa **B** a függőségi halmazból lezártak segítségével levezetve (házi feladat).

Mivel az egyetlen nem triviális függés bal oldalán legalább szerepel a **B** attribútum, így az R_{221} reláció BCNF-ben van.

Az R_{222} kulcsa **AB** a függőségi halmazból lezártak segítségével levezetve (házi feladat).

Mivel az egyetlen nem triviális függés bal oldalán nem szerepel legalább az **AB** attribútumhalmaz, ezért az R_{222} reláció nincs BCNF-ben, folytatnunk kell a dekomponálását $A \rightarrow C$ függés mentén.

$R_{2221}(A, C)$; $F_{2221}=\{A \rightarrow C\}$

$R_{2222}(A, B)$; $F_{2222}=\{\}$

Az R_{2221} kulcsa **A** a függőségi halmazból lezártak segítségével levezetve (házi feladat).

Mivel az egyetlen nem triviális függés bal oldalán legalább szerepel a **A** attribútum, így az R_{2221} reláció BCNF-ben van.

Az R_{2222} kulcsa **AB**, mivel csak triviális függéseket tartalmaz.

Mivel az egyetlen nem triviális függés sincs a relációban így az R_{2221} reláció BCNF-ben van.

Nézzük, mely relációkra bontottuk végül a kiinduló R (<u>A</u>, <u>B</u>, C, D, E, F) relációt az $F=\{A \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow F\}$ függőségi halmaz elemei mentén:

$R_1(C, E)$; $F_1=\{C \rightarrow E\}$
$R_{21}(D, F)$; $F_{21}=\{D \rightarrow F\}$
$R_{221}(B, D)$; $F_{221}=\{B \rightarrow D\}$
$R_{2221}(A, C)$; $F_{2221}=\{A \rightarrow C\}$
$R_{2222}(A, B)$; $F_{2222}=\{\}$

Veszteségmentes a felbontás? A relációkat a táblában a fenti sorrendjük alapján újra számozzuk, hogy a táblázat kitöltésénél ne okozzon kavarodást az indexálásban, tehát az R_{21} az R_2 lesz, R_{221} az R_3 lesz, stb.
A veszteségmentességet az $F=\{A \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow F\}$ funkcionális függőség halmaz alapján az ismert Chase algoritmussal ellenőrizhetjük:

	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>	<u>E</u>	<u>F</u>
R_1	b_{11}	b_{12}	a_3	b_{14}	a_5	b_{16}
R_2	b_{21}	b_{22}	b_{23}	a_4	b_{25}	a_6
R_3	b_{31}	a_2	b_{33}	a_4	b_{35}	b_{36}
R_4	a_1	b_{42}	a_3	b_{44}	b_{45}	b_{46}
R_5	a_1	a_2	b_{53}	b_{54}	b_{55}	b_{56}

$A \rightarrow C$-t alkalmazva b_{53} egyenlővé válik a_3-al:

	A	B	C	D	E	F
R_1	b_{11}	b_{12}	a_3	b_{14}	a_5	b_{16}
R_2	b_{21}	b_{22}	b_{23}	a_4	b_{25}	a_6
R_3	b_{31}	a_2	b_{33}	a_4	b_{35}	b_{36}
R_4	a_1	b_{42}	$\mathbf{a_3}$	b_{44}	b_{45}	b_{46}
R_5	a_1	a_2	$\mathbf{a_3}$	b_{54}	b_{55}	b_{56}

$C \rightarrow E$-t alkalmazva b_{45} és b_{55} egyenlővé válik a_5-el:

	A	B	C	D	E	F
R_1	b_{11}	b_{12}	a_3	b_{14}	$\mathbf{a_5}$	b_{16}
R_2	b_{21}	b_{22}	b_{23}	a_4	b_{25}	a_6
R_3	b_{31}	a_2	b_{33}	a_4	b_{35}	b_{36}
R_4	a_1	b_{42}	a_3	b_{44}	$\mathbf{a_5}$	b_{46}
R_5	a_1	a_2	a_3	b_{54}	$\mathbf{a_5}$	b_{56}

$B \rightarrow D$-t alkalmazva b_{54} egyenlővé válik a_4-el:

	A	B	C	D	E	F
R_1	b_{11}	b_{12}	a_3	b_{14}	a_5	b_{16}
R_2	b_{21}	b_{22}	b_{23}	a_4	b_{25}	a_6
R_3	b_{31}	a_2	b_{33}	$\mathbf{a_4}$	b_{35}	b_{36}
R_4	a_1	b_{42}	a_3	b_{44}	a_5	b_{46}
R_5	a_1	a_2	a_3	$\mathbf{a_4}$	a_5	b_{56}

$D \rightarrow F$-t alkalmazva b_{36} és b_{56} egyenlővé válik a_6-al:

	A	B	C	D	E	F
R_1	b_{11}	b_{12}	a_3	b_{14}	a_5	b_{16}
R_2	b_{21}	b_{22}	b_{23}	a_4	b_{25}	$\mathbf{a_6}$
R_3	b_{31}	a_2	b_{33}	a_4	b_{35}	$\mathbf{a_6}$
R_4	a_1	b_{42}	a_3	b_{44}	a_5	b_{46}
R_5	a_1	a_2	a_3	a_4	a_5	$\mathbf{a_6}$

Ha megnézzük a táblázatot, akkor azt látjuk, hogy az R_5-ös sorban csupa a_k-ból álló érték szerepel, tehát **a felbontás veszteségmentes.**

Nézzük meg, hogy az eredeti funkcionális függőségi halmaz alapján $F=\{A \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow F\}$ függőségőrző lett-e a felbontás?

$F_1=\{C \rightarrow E\}$

$F_{21}=\{D \rightarrow F\}$

$F_{221}=\{B \rightarrow D\}$

$F_{2221}=\{A \rightarrow C\}$

$F_{2222}=\{\}$

$F_1 \cup F_{21} \cup F_{221} \cup F_{2221} \cup F_{2222} \equiv F$?

$F_1 \cup F_{21} \cup F_{221} \cup F_{2221} \cup F_{2222} = \{A \rightarrow C, B \rightarrow D, C \rightarrow E, D \rightarrow F\} \equiv F$, tehát **a felbontás függőségőrző**!

Gyakorló feladatok:

Adott R (A, B, C, D, E, F) reláció és $F=\{A \rightarrow B, B \rightarrow D, C \rightarrow F\}$ függőségi halmaz.
Készítsük el a reláció veszteségmentes és függőségőrző felbontását BCNF-re

Adott R (A, B, C, D, E) reláció és $F=\{E \rightarrow A, B \rightarrow D\}$ függőségi halmaz.
Készítsük el a reláció veszteségmentes és függőségőrző felbontását BCNF-re

Összefoglalás:

Ugyanazon a példán keresztül megnéztük, miért kaptunk különböző megoldásokat, miért lett az egyik megoldás nem függőségőrző.
A BCNF-re történő dekomponálás meghatározott lépései jól működnek, veszteségmentes felbontáshoz vezetnek, de figyelnünk kell arra, hogy a tranzitív függések esetén a második tag szerint kezdjük a felbontást, ha függőségőrző is legyen a felbontás.
Minden esetben érdemes a veszteségmentességet ellenőrizni a Chase algoritmus segítségével, valamint a függőségőrzést a felbontott relációk függőségi halmazainak uniójának képzésével.

Az adatbázistervezés célja

BCNF-re hozás:

- Veszteségmentes felbontás
- Függőségőrző felbontás

Ha ez nem valósítható meg, akkor:

3NF-re hozás:

- Veszteségmentes felbontás
- Függőségőrző felbontás

Ha BCNF-ben nem tudjuk biztosítani a függőségőrzést, akkor a 3NF-re bontást válasszuk, mert így mind a veszteségmentesség, mind a függőségőrzés biztosítható.

VII. A relációs lekérdező nyelvek

A lekérdező nyelvek: az adatmanipulációt és az adatok elérését teszik lehetővé az adatbázisból.

A relációs modell lekérdező nyelveire jellemző:

- Nagy adathalmazok egyszerű és hatékony kezelése és lekérdezése
- Formálisan matematikailag megalapozott
- Automatikus lekérdezés optimalizálás

A relációs modellre vonatkozó előismeretek:

- Midkét konvenció használható (az SQL-ben is)
- A lekérdezéseket relációpéldányokra alkalmazzuk és az eredmény szintén relációpéldány
- A lekérdezés input relációsémái illetve az adott lekérdezés eredményének a sémája rögzített

Példa relációpéldányokra:

TAG

AZON	NEV	IRSZ	VAROS	UTCA	SZUL_DAT
0524	Kovács Zoltán	4028	Debrecen	Kút u.32.	12-AUG-46
0525	Tar Ede	4090	Polgár	Kerek u.96.	03-JAN-40
0526	Villám Éva	4029	Debrecen	Kassai u.55.	22-JAN-70
0527	Kiss Zoárd	3508	Miskolc	Búza tér 3.	05-FEB-72
0528	Felhő Katalin	4183			
0529	Nagy Péterné	4024	Debrecen	Csap u.11.	26-OCT-40
0530	Szekeres Endre	4027	Debrecen	Füredi u.33.	11-MAR-74
0531	Tölgyes Emese	5000	Szolnok	Fő ut.5.	14-FEB-60

KOLCSON

AZON	KOD	KOLCS_IDO	KOLCS_DAT	STATUSZ
0524	0839	1	09-NOV-97	B
0524	0842	3	14-NOV-97	B
0526	0839		15-NOV-97	
0529	0839	4	23-NOV-97	B
0530	0839		27-NOV-98	
0531	0839	1	03-DEC-98	
0530	0842	3	10-DEC-98	
0526	0842		03-JAN-99	
0527	0838	3	03-FEB-99	B
0529	0842	1	16-JUN-99	B

CD

KOD	CD_CIM	ELOADO	KIAD_EV	BESZ_AR
0838	Kisértés	Tátrai Band	1992	1000
0839	Ringasd el magad	LGT	1990	
0840	Mindenki	LGT	1992	
0841	Mindig magasabbra	LGT	1994	1050
0842	Edda 13	Edda	1992	1000

Két matematikai lekérdező nyelv

az alapja a konkrét lekérdező nyelveknek:

1. **A relációs algebra:**
- halmazorientált
- algebrai eszközökkel dolgozik
- procedurális: műveletekkel adom meg (hogyan?)

2. **A relációs kalkulusok** (rekord-, tartomány-alapú)
- a matematikai logikán alapul
- deklaratív nyelv: feltételekkel adom meg (mit?)
-

Konkrét implementációik vannak:
- Relációs algebrán (RALG) alapul: az ISBL
- Rekord alapú relációs kalkuluson (TRC): a QBE
- A fenti kettőn (RALG, TRC) alapul: az SQL
- Tartomány alapú relációs kalkuluson: a QUEL

VIII. A relációs algebra

A relációs algebra sikeres elsajátítása előkészít minket arra, hogy az SQL nyelven könnyebben fogalmazzuk meg a lekérdezéseinket, ezért fontos a megtanulása, gyakorlása.

A relációs algebra alapműveletei

- Szelekció (σ) sorok kiválasztása
- Projekció (π) oszlopok kiválasztása
- Átnevezés (ρ) oszlopnevek átnevezése
- Descartes szorzat (×) két reláció kombinálása
- Unió (∪) két reláció összes sora
- Különbség (–) az 1. reláció sorai, de a 2.-é nem

További fontos műveletek

- Összekapcsolások és a természetes összekapcsolás
- Hányados a "minden" kifejezésére
- Metszet két reláció közös sorai
- Külső összekapcsolások NULL-al kiegészül

Minden művelet eredménye reláció ("zártság")

A relációs algebra alapműveletei

Szelekció: $\sigma_{feltétel}$ (rel_név)

Egy adott reláció rekordjainak azon részhalmazát adja, amelybe pontosan a szelekciós feltételt teljesítő rekordok tartoznak bele.

A szelekciós feltétel: egy logikai kifejezés, melyben

- Az operandusok aritmetikai összehasonlító és logikai műveleti jelek
- Az operátorok (formálisan) attribútumok és konstansok

Az eredményül kapott reláció

- fokszáma = a kiindulási reláció fokszámával
- rekordok száma ≤ mint az induló rekordok száma

Az eredményreláció inputja lehet egy további relációs algebrai műveletnek.

Két jelölésmód is használható:

Példa: $\sigma_{VAROS = "Debrecen" \text{ or } IRSZ = "3508"}$ (TAG)

$\sigma_{\$4 = "Debrecen" \text{ or } \$3 = "3508"}$ (TAG)

AZON	NEV	IRSZ	VAROS	UTCA	SZUL_DAT
0524	Kovács Zoltán	4028	Debrecen	Kút u.32.	12-AUG-46
0526	Villám Éva	4029	Debrecen	Kassai u.55.	22-JAN-70
0527	Kiss Zoárd	3508	Miskolc	Búza ter 3.	05-FEB-72
0529	Nagy Péterné	4024	Debrecen	Csap u.11.	26-OCT-40
0530	Szekeres Endre	4027	Debrecen	Füredi u.33.	11-MAR-74

Tulajdonságai:

- A szelekció kommutatív művelet:

$\sigma_{f1}(\sigma_{f2}(R)) = \sigma_{f2}(\sigma_{f1}(R))$

- Szelekciósorozat mindig értelmezhető egyetlen szelekcióként:

$\sigma_{f1}(\sigma_{f2} \ldots (\sigma_{fn}(R)) \ldots) = \sigma_{f1 \text{ and } f2 \text{ and } \ldots \text{ and } fn}(R)$

Példa:
Adott R(A, B) reláció a következő értékekkel:

A	*B*
1	2
3	4

$\sigma_{A < 2}(R)$ művelet eredménye:

A	*B*
1	2

$\sigma_{A < 2 \text{ and } B > 3}(R)$ művelet eredménye (üres tábla):

A	*B*

Projekció: $\pi_{attributum\_lista}$ (rel_név)

Az eredményrelációba csak a kiválasztott attribútumokhoz tartozó értékek kerülnek.
Az attribútumlistán az attribútumok sorrendje tetszőleges, de a keletkezett reláció attribútumainak sorrendjét meghatározza ez a sorrend.

Az eredményül kapott reláció

- sémáját (fokszámát) a kiindulásban felsorolt attribútumok (száma) határozza meg.
- rekordszáma: Ha a kiválasztott attribútumok között van kulcs, akkor az induló rekordok száma egyenlő az eredményrekordok számával. Ha nem szerepel kulcs, akkor az eredményrekordok száma kevesebb lehet, mint az induló rekordok száma (mivel a projekció kiküszöböli a duplikálást).

Megjegyzés: A valós rendszerek nem küszöbölik ki a duplikációt, csak ha a felhasználó explicit kéri azt.

Két jelölésmód is használható:

Példa: $\pi_{NEV, IRSZ, VAROS}(TAG)$

$\pi_{\$2, \$3, \$4}(TAG)$

NEV	IRSZ	VAROS
Kovács Zoltán	4028	Debrecen
Tar Ede	4090	Polgár
Villám Éva	4029	Debrecen
Kiss Zoárd	3508	Miskolc
Felhő Katalin	4183	Kaba
Nagy Péterné	4024	Debrecen
Szekeres Endre	4027	Debrecen
Tölgyes Emese	5000	Szolnok

Tulajdonságai:

- Projekciósorozat, ha értelmezhető, akkor egyetlen projekcióként fogható fel:

$\pi_{l1}(\pi_{l2}(R)) = \pi_{l1}(R)$

ez csak akkor értelmezhető, ha $l2 \supseteq l1$.

Példa:

Adott R(A, B) reláció a következő értékekkel:

A	*B*
1	2
3	4

$\pi_{B}(R)$ művelet eredménye:

B
2
4

$\pi_{B, A}(R)$ művelet eredménye:

B	*A*
2	1
4	3

Átnevezés: $\rho_{ÚJONÉV \leftarrow ONÉV}$ *(rel_név)*

Ez tulajdonképpen nem reláció algebrai művelet, hanem technikai dolog, az oszlopok átnevezését értjük rajta (pl. adatbázisban az oszlopnevek más nyelvű, illetve ékezetes változatának megmutatására (lásd később az SQL részben).

Csak a névvel ellátott jelölésmódban használjuk. Példa: $\rho_{TAGNEV \leftarrow NEV}(TAG)$

Megjegyzés: A valós rendszerekben, pl. az SQL-ben az oszlopokat lehet minősíteni a relációnévvel (pl. NEV helyett TAG.NEV, így nincs szükség az oszlopok átnevezésére, a továbbiakban ezt az egyszerűbb jelölést használjuk).

Megjegyzés: A valós rendszerekben, pl. az SQL-ben lehetőség van a relációk átnevezésére, illetve hivatkozási (alias) névvel láthatjuk el a táblákat, mezőket.

Halmazműveletek

A relációkat rekordok halmazának tekintjük, és két reláción végrehajthatjuk a szokásos halmaz műveleteket, eredményként egy relációt kapunk.

- **Unió: R∪S**
- **Kivonás: R−S**
- **Metszet: R∩S**

A szokásos halmazelméleti értelemben véve.

Az ∪, ∩ és − műveleteknél a két relációnak **unió-kompatibilisnek** kell lennie, vagyis

- A két reláció fokszáma azonos legyen
- Az egyes helyeken lévő megfelelő elemek ugyanazon tartományból legyenek (tehát csak azonos szerkezetű, vagy azonos szerkezetűre hozott (pl. projekcióval) relációkon hajtható végre!)

A Metszet kifejezhető a relációs algebrai alapműveletekkel:
$R \cap S = R - (R - S)$

Tulajdonságai:
A halmazműveletekre megszokott tulajdonságok, és az ismert de Morgan azonosságok,
pl. $R \cap S = S \cap R$,
$R \cap (S \cap T) = (R \cap S) \cap T$
$R \cup (S \cap T) = (R \cup S) \cap (R \cup T)$
$R \cap (S \cup T) = (R \cap S) \cup (R \cap T)$, stb.

Példa:
Adott R(A, B) és S(A, B) reláció a következő értékekkel:

R

A	*B*
1	2
3	4

S

A	*B*
3	4
5	6

Végezzük el a következő halmazműveleteket:

R∩S

A	*B*
3	4

R∪S

A	*B*
1	2
3	4
5	6

R–S

A	*B*
1	2

S–R

A	*B*
5	6

Összetett relációsalgebrai műveletek:

Természetes összekapcsolás,

jele: ⋈

Már korábban átbeszéltük, de szerepeljen itt is példával alátámasztva.
Természetes összekapcsolás művelete akkor végezhető el két reláción, ha vannak közös attribútumaik, melyek típusban és hosszban is megegyeznek.
Eredményként azokat a sorokat kapcsolja össze, melyek értéke a közös attribútumban megegyezik.

Példa: Adott R(A, B) és S(B, C, D) reláció. Végezzük el a természetes összekpacsolás műveletét

R

A	*B*
1	2
3	4

S

B	*C*	*D*
2	5	6
4	7	8
9	10	11

R ⋈ S

A	*B*	*C*	*D*
1	2	5	6
3	4	7	8

Mivel előfeltétel, hogy legyen közös attribútuma a két relációnak, melyeket a műveletet elvégezzük és érték szerinti azonos sorokat kapcsolja össze, ezért az eredményrelációban a közös attribútum és annak értéke csak egyszer szerepel.

Példa: Adott U(A, B, C) és U(B, C, D) reláció. Végezzük el a természetes összekpacsolás műveletét

U

A	*B*	*C*
1	2	3
6	7	9
9	7	8

V

B	*C*	*D*
2	3	4
2	3	5
7	8	10

U⋈V

A	*B*	*C*	*D*
1	2	3	4
1	2	3	5
9	7	8	10

Descartes szorzat

jele: ×

ha $R(A_1, ... , A_m) \times S(B_1, ... , B_n)$

- Az eredmény fokszáma m+n
- A rekordok száma: R-ben: r db, S-ben: s db, az eredményben: r × s db.

A descartes szorzat a bal oldali reláció minden sorához hozzárendeli a jobb oldali reláció minden sorát.

Példa:

R

A	*B*
1	2
3	4

S

B	*C*	*D*
2	5	6
4	7	8
9	10	11

R×S

A	*R.B*	*S.B*	*C*	*D*
1	2	2	5	6
1	2	4	7	8
1	2	9	10	11
3	4	2	5	6
3	4	4	7	8
3	4	9	10	11

Descartes szorzatnál amennyiben vannak azonos mezőnevek a műveletben résztvevő táblákban, akkor azon nevét ki kell egészíteni a tábla nevével, így egyértelműsítve, melyik táblához tartozó értéket képviselnek.

Tulajdonságai:
A halmazműveletekre megszokott tulajdonságok,
pl. $R \cup S = S \cup R$, $R - S \neq S - R$
$R \times (S \times T) = (R \times S) \times T$, stb...

Théta összekapcsolás

jele: $R \times_{felt} S$

ahol a feltétel = f_1 and f_2 and ... and f_n,a részfeltételek: $f_i = A_i \Theta B_j$ alakú aritmetikai összehasonlítások, és a $\Theta \in \{<, >, \leq, \geq, =, \neq\}$

Az eredményül kapott reláció
- fokszáma = m+n lesz (séma, mint a Descartes szorzatnál)
- a rekordok száma maximum r × s lehet, illetve a feltételtől függően kevesebb.
-

Kifejezhető a relációs algebrai alapműveletekkel:
$R \times_{felt} S = \sigma_{felt} (R \times S)$

Egyen-összekapcsolás (Equijoin):

A fenti Théta-összekapcsolás speciális esete, ahol a Θ az = egyenlőség, vagyis valamely oszlopértékek a másikkal páronként megegyeznek.

Hányados

R÷S
A "minden" kifejezésére szolgál az algebrában.
Hányados előfeltétele: S attribútumai elő kell hogy forduljanak R attribútumai között: $Y \subseteq X \cup Y$.
Legyenek R(X,Y), S(Y) és T(X) relációsémák.

Definíciója: Az R és S relációk hányadosaként kapott $T=R \div S$ reláció az a legnagyobb számosságú (legtöbb rekordból álló) reláció, amelyre $T \times S \subseteq R$.

Jelentése: Más szóval ez azt jelenti, hogy a T reláció tartalmazza az összes olyan t rekordot, melyre igaz, hogy **minden** egyes S-beli s rekordra a ts rekord benne van az R relációban.

Kifejezhető a relációs algebrai alapműveletekkel:
Az eredményül kapott reláció sémája: R azon attribútumaiból áll, amelyek nincsennek benne az S attribútumai között, vagyis T(X).
Az eredményül kapott relációpéldány pedig:
Legyen $T_1 = \pi_X(R)$ és $T_2 = \pi_X((T_1 \times S) - R)$.
Ekkor $T = T_1 - T_2$.

Példa:

R(A, B)

a_1	b_1
a_2	b_1
a_3	b_1
a_4	b_1
a_1	b_2
a_3	b_2
a_2	b_3
a_3	b_3
a_4	b_3
a_1	b_4
a_2	b_4
a_3	b_4

S(A)

a_1
a_2
a_3

T(B)

b_1
b_4

Külső összekapcsolások

Egyen-összekapcsolás

Az eredményében akkor szerepel egy sor, ha a párja szerepel a másik relációban.

Outer-join (külső összekapcsolás)

a hiányzó párok is szerepelnek az eredményrelációban, és a hiányzó mezők NULL értékkel vesznek részt az összekapcsolásban.
Ez alapján három féle lehet:

- baloldali
- jobboldali
- mindkét oldali

pl.: Olvasó tagja már a könyvtárnak, de még nem kölcsönzött ki könyvet még.

Relációs algebrai kifejezések

$e(X_1, \ldots, X_n)$

1) **Operandusai:** $X_1, \ldots, X_n$ változók. Az R, Q, S, … relációs konstansok relációs algebrai kifejezések.

2) **Alapműveletek:** Ha e_1 és e_2 relációs algebrai kifejezések, akkor $\sigma_{\text{feltétel}}(e_1)$, $\pi_{\text{lista}}(e_1)$, $\rho_{B \leftarrow A}(e_1)$, $e_1 \times e_2$, $e_1 \cup e_2$, $e_1 - e_2$ is rel.algebrai kifejezések.

3) **Zárójelek:** Ha e relációs algebrai kifejezés, akkor (e_1) is relációs algebrai kifejezés.

4) Ezek és csak ezek a relációs algebrai kifejezések.

Műveletek priritási sorrendje:

1.) ( ) - zárójelek
2.) unér-műveletek: σ, π, ρ
3.) multiplikatív műveletek: $\times$, $\bowtie$
4.) additív műveletek: $\cap$, $\cup$, $-$

Ekvivalencia

Jelölés: Legyen **$e(X_1, \ldots, X_n)$** relációs algebrai kifejezés és $R_1, \ldots, R_n$ adott relációk.
Ekkor **$e(R_1, \ldots, R_n)$** azt a relációt jelenti, amelyet úgy kapunk, hogy relációs algebrai kifejezésben minden i-re X_i-t R_i-vel helyettesítve és elvégezzük az relációs algebrai kifejezésben szereplő műveleteket.

Ekvivalencia: Akkor mondjuk, hogy két relációs algebrai kifejezés **$e_1(X_1, \ldots, X_n)$** és **$e_2(X_1, \ldots, X_n)$** ekvivalens, ha minden $R_1, \ldots, R_n$ reláció esetén

- vagy **$e_1(X_1, \ldots, X_n) = e_2(X_1, \ldots, X_n)$**, ugyanazt a relációt adják,
- vagy mindkettő nem-definiált (a kiértékelés során valamely művelet nem értelmezhető).

Relációs algebrai azonosságok
Igazolja a relációs algebrai műveletekre érvényes azonosságokat!
Például:

- A szelekció, és a természetes összekapcsolás kommutatív és asszociatív.
- Milyen feltételekkel cserélhető fel a szelekció és a projekció?
- Milyen feltételekkel cserélhető fel a szelekció és a természetes összekapcsolás?
- Milyen feltételekkel cserélhető fel a projekció és a természetes összekapcsolás?

A relációs műveletek teljessége és függetlensége

- Melyek azok a műveletek, amelyek az eddig említett műveletek segítségével kifejezhetők? (pl. a $\{\sigma, \pi, \cup, -, \times\}$ -ből mind, és hogyan?)
- Igazoljuk, hogy véges relációk tranzitív lezárási művelete nem fejezhető ki relációs algebrában!
- Igazoljuk, hogy a relációs algebrai alapműveletek függetlenek, azaz egyik művelet elhagyása után sem fejezhető ki a szóban forgó művelet a többiből (minden alapműveletnek emelje ki olyan jellemző tulajdonságát, ami miatt a többiből nem fejezhető ki, pl. a projekció csökkenti az oszlopok számát, a többi egyik sem, a különbség nem monoton művelet, de a többi az.)

Feladatok lekérdezések kifejezésére

Fejezze ki relációs algebrai kifejezésként az alábbi lekérdezéseket:

- Keressük azokat az együtteseket, akik két különböző CD-t adtak ki ugyanabban az évben!
- Kik azok, akik LGT CD-t kölcsönöztek?
- Kik azok, akik minden LGT CD-t kölcsönöztek?
- Kik azok, akik legalább azokat a CD-ket kikölcsönöztek, amit Kovács Zoltán?
- Kik azok, akik csak olyan CD-t kölcsönöztek ki, amit Kovács Zoltán is kivett?
- Kik azok, akik pontosan azokat a CD-ket kölcsönözték ki, amiket Kovács Zoltán?

Gyakorló feladatok relációs algebrára

A relációs algebra begyakorlásához idő és befektetett energia kell. Kérem az alábbi feladatok megoldását gondolják végig, valóban a kívánt eredményt adják-e, illetve az összetett feladatoknál próbáljanak meg egyéni megoldást is találni. Sokszor egy adott feladat többféle képpen is megoldható.

Első feladatsorozat egy relációval:

Relációs séma: Szeret (név, gyümölcs), sz előfordulás

Feladatok:

1.a. Mit szeret Józsi?
A = Πgyüm (σnév='Józsi' (sz))

1.b. Mit nem szeret Józsi? Πgyüm (sz) - A

2.a. Ki szereti a narancsot?
B = Πnév (σgyümölcs='Narancs' (sz))

2.b. Ki nem szereti a narancsot? Πnév (sz) - B

3. Ki szeret minden gyümölcsöt?
Πnév (sz) - Πnév {Πnév (sz) x Πgyümölcs (sz)- sz}

4. Ki szeret legalább két gyümölcsöt?
Π$1 (σ$2#$4 és $1=$3 (sz1 x sz2)) ; legalább három gyümölcsöt: sz1 x sz2 x sz3

5. Ki szeret legfeljebb két féle gyümölcsöt? Összes-legalább 3-at szeret

6. Ki szeret pontosan kétféle gyümölcsöt? legalább két ∩ legfeljebb két gyümölcsöt szeret

7. Ki szereti legalább azokat a gyümölcsöket, mint Béla?

Πnév (sz) - Πnév { Πnév (sz) x Πgyümölcs (σnév='Béla' (sz)) -sz }

8. Ki szereti legfeljebb azokat a gyümölcsöket, mint Béla?
(név- legalább 1-et, amit Béla nem szeret)

Πnév (sz) - Πnév {sz - Πnév (sz) x Πgyümölcs (σnév='Béla' (sz))}

9. Ki szereti pontosan azt, mint Béla? legalább azt szereti ∩ legfeljebb azt szereti

10. Kik azok, akik különböző gyümölcsöket szeretnek?

sz1 = sz2, k = sz1 x sz2, k ' = Π$1$4$3$2 (k) Kül = Πsz1.név, sz2.név (k - k ')

11. Kik szeretik pontosan u.a. a gyümölcsöket? Πnév (sz) x Πnév (sz) -Kül. gyüm.

Második feladatsorozat három relációval:

Relációs sémák: Szeret (név, sör), **sz** előfordulás
Látogat(név, bár), **L** előfordulás
Ad(bár, sör)

L $\bowtie$ ad = mit tud inni
L $\bowtie$ sz = mely bárokban mit kéne adniuk
ad $\bowtie$ sz = milyen bárba érdemes járnia

Feladatok:

1. Kik azok, akik legalább egy olyan bárba járnak, ahol legalább egy olyan sört adnak, amit szeretnek?

$\Pi_{\text{név}}$ (sz $\cap \Pi_{\text{név, sör}}$(L $\bowtie$ad))

2. Kik azok, akik csak olyan bárba járnak, ahol felszolgálják legalább egy kedvenc sörét?

A = $\Pi_{\text{név}}$ (L) $\cap$ $\Pi_{\text{név}}$ (sz)- $\Pi_{\text{név}}$ (L - $\Pi_{\text{név,bár}}$ (sz $\bowtie$ad))

(jár jó helyre is - jár nem jó helyre is)

3. Kik azok, akik csak olyan bárba járnak, ahol minden kedvencük kapható?

A - $\Pi_{\text{név}}$ ((L $\bowtie$sz) - (L $\bowtie$ad $\bowtie$sz))
(jár, szereti, de nem adják)

4. Kik azok, akik nem járnak olyan bárokba, ahol felszolgálnának olyan sört, amit ők szeretnek?

($\Pi_{\text{név}}$ (sz)$\cup\Pi_{\text{név}}$ (L) - A

Összefoglalás (A relációs algebra)

- A relációs algebra alapműveletei: szelekció, projekció, átnevezés, és a halmazműveletek (Descartes szorzat, unió, különbség)
- További műveletek: metszet, összekapcsolások, hányados, külső összekapcsolások
- Lekérdezések kifejezése relációs algebrában, relációs algebrai kifejezések, műveletek prioritása, ekvivalencia, relációs algebrai azonosságok, relációs műveletek teljessége és függetlensége.
- Feladatok lekérdezések kifejezésére relációs algebrában

IX. SQL Structured Query language

Strukturált lekérdező nyelv, valójában több ennél; egy teljes adatbázis kezelő nyelv, melyet ma majdnem szabványnak tekinthetünk.
Formalizmusát tekintve az SQL a relációs kalkulushoz, a nem procedurális nyelvekhez áll közel. A felhasználó lényegében csak azt mondja meg, mit akar [6]. Azt, hogy ez hogyan, milyen lépések alapján történjen meg, a nyelv értelmezője, fordítóprogramja fogja meghatározni az adatbázis szerkezetének ismeretében. A feladat megoldási algoritmusának meghatározásához jó SQL rendszerek a logikai és fizikai adatszerkezeten kívül figyelembe veszik a tényleges adattartalmat és adatmennyiséget, hogy az elérési időt optimalizálják. Az ehhez szükséges információkat a rendszer katalógusából veszik.

Története:
Az 1990-es évek elején kezdett megjelenni és hatékonyságának köszönhetően gyorsan támogatottá vált a különböző adatbáziskezelő-rendszerekben. Korábban az adatbáziskezelő-rendszerek saját programnyelvvel rendelkeztek, mely segítségével a különböző lekérdezéseket lehetett megfogalmazni, hatékonyságuk pedig a rendszer hatékonyságától függött. Első lépésként csak támogatni kezdték az adatbáziskezelő-rendszerek az SQL alapú lekérdezés-megfogalmazást a saját programnyelvük mellett, így összehasonlítva őket egymással az SQL meg tudta mutatni a hatékonyságát. Ma már önálló szerverszolgáltatásként használhatjuk és nagy méretű adatbázisokat szolgál ki.
Eredetileg az Oracle, az IBM és a Microsoft együttműködése határozta meg a nyelv fejlődését, mely újabb és újabb lehetőségeket, illetve funkciókat jelentett azonos módon megoldva, használva, így közel mindegy volt, hogy mely cég SQL szolgáltatását szeretnénk használni, azonos módon kellett megfogalmazni a kérést. Idővel kikerült a nyelv felügyelete az Egyesült Államok területéről, melynek köszönhetően már nincs megfelelő együttműködés a különböző cégek között, a nyelv fejlődése az adott cég által megfogalmazott elvárások szerint alakul, ennek köszönhetően már lényeges különbségek vannak az egyes cégek SQL lekérdező nyelve által kínált szolgáltatásoknak és azok elérésének módja között. Ennek következtében már cégspecifikus az adott lekérdezés megfogalmazása.

Az SQL utasításokat 4 csoportra osztjuk:
- adatleíró utasítások (adatdefiníciós nyelv, Data Definition Language=DDL)
- adatkezelő utasítások (adatmanipulációs/adatmódosító nyelv, Data Manipulation Language=DML)
- vezérlő utasítások
- egyéb utasítások

Az **adatleíró utasítások** közé tartoznak az adatbázis szerkezetét leíró és módosító utasítások.
Legfontosabbak közülük a

- Create (létrehoz)
- Drop (megszüntet)
- Alter (módosít)

típusú utasítások.

Az **adatkezelő utasítások** meglévő táblázatokból, nézetekből választanak ki, törölnek, módosítanak meglévő sorokat, új sorokat írnak le.
Legfontosabbak közülük a

- Select (kiválaszt)
- Delete (töröl)
- Update (módosít)
- Insert (beír)

típusú utasítások.
Ezek még kiegészülnek a programnyelvbe való beillesztéshez szükséges további utasításokkal.

A **vezérlő utasítások** segítségével engedélyezhetjük különböző SQL objektumok használatát, biztosíthatjuk több lépésből álló utasításcsoport konisztens végrehajtását, vagy bármely hiba esetén a kiindulási állapotba való visszaállítást.
Legfontosabbak közülük a
Grant (jogosultság megadása)
Revoke (jogosultság visszavonása)
Commit (utasítások eredményének véglegesítése)
Rollback (eredeti állapot visszaállítása)
Lock (adatbázis objektumok zárolása)
Whenever (hibakezelés programban)
Set (rendszerparaméterek beállítása)
típusú utasítások.

Adatleíró utasítások:

A táblázatok, indexek, nézetek létrehozása, megszüntetése minden SQL-en alapuló adatbázis-kezelő rendszerben vannak megfelelő utasítások, viszont ezek módosítását már nem mindegyik engedi meg. Bármilyen objektumot csak olyan felhasználó hozhat létre, módosíthat, törölhet, akinek erre jogosultsága van (pl. adatbázis-felügyelő, vagy akiknek ő erre jogot adott). Általában az objektum létrehozója jogosult a módosításra és a törlésre is. Más felhasználók csak akkor tehetik meg, ha ezt számukra az objektum tulajdonosa megengedte.

Táblázatok létrehozása (Create Table utasítás)
A táblázat létrehozásához meg kell adnunk a táblázat nevét és oszlopainak pontos definícióját. Ezenkívül megadhatjuk, hogy kit kell a táblázat tulajdonosának tekinteni, illetve az adatbázison belül mely területen helyezkedik el.

CREATE TABLE táblázatnév(oszlopdefiníciók
oszlopnév adattípus NOT NULL
NOT NULL WITH DEFAULT
PRIMARY KEY (oszlopnév))
IN adatbázisterület

ahol
oszlopdefiníció: oszlopnév adattípus NOT NULL
NOT NULL WITH DEFAULT

Adatípus: CHARACTER(n) n karakterből álló jelsorozat (általában n<256)
INTEGER 4 byte-on ábrázolt bináris egész szám
DECIMAL (p,q) összesen p jegyből álló decimális szám. q db tizedes jeggyel

NOT NULL megadása esetén ennek az oszlopnak mindenképpen értéket kell adni adatbevitelkor. Ha nem adjuk meg és nem viszünk be adatot, akkor az oszlop értéke NULL érték lesz, vagy ha a NOT NULL DEFAULT kiegészítést adjuk meg, akkor az alapértelmezés (nulla, szóköz).

Pl. CREATE TABLE gepjármu
(rendszam CHARACTER(6) NOT NULL,
gyartmany CHARACTER(16) NOT NULL,
tipus CHARACTER(16),
szín CHARACTER(12) NOT NULL,
urtartalom INTEGER NOT NULL,
PRIMARY KEY (rendszam))
IN gepjárműnyílvántartás

Index létrehozása (CREATE INDEX utasítás)
Index létrehozásánál meg kell adni az index nevét, valamint azt, hogy melyik táblázat melyik oszlopaira épüljön fel. Ezenkívül megadhatjuk, hogy az index-oszlopok értéke a táblázatban egyedi legyen-e (UNIQUE), növekvő, vagy csökkenő.
CREATE [UNIQUE] INDEX indexnév ON táblázatnév (oszlopnév1 [ASC][DESC], ...)

Egy index általában max. 16 oszlopból állhat és az oszlopok együttes hossza is korlátozott (rendszerint120-254 byte között) Általában olyan oszlopra, amelyik index, vagy index része, ne engedjünk meg NULL értéket.

Táblázat, index megszüntetése (DROP utasítás)
DROP INDEX indexnév
DROP TABLE táblázatnév
A táblázat megszüntetésekor automatikusan törlődik a táblázat összes adata, indexe.

Adatkezelő utasítások

Ezek segítségével a meglévő táblázatokból választhatunk ki, írhatunk be, módosíthatunk, vagy törölhetünk sorokat.
Az utasítást csak az hajthatja végre, akinek az abban szereplő táblázatokra vonatkozóan az adott utasítás elvégzésére jogosultsága van.

Szelektálás (SELECT utasítás)
Meghatározott feltételeknek eleget tevő adatokat választunk ki 1 vagy több táblázatból. Az utasítás eredménye mindig egy ideiglenes táblázat.

SELECT [ALL,DISTINCT] oszlopnév1, vagy kifejezés1, vagy SQL függvény1,... (*)
FROM táblázatnév
WHERE kiválasztási feltétel
GROUP BY oszlopnév1,...
HAVING csoportkiválasztási feltétel
ORDER BY oszlopnév1 [ASC,DESC], ...

ALL minden feltételnek megfelelő sort kiválasztunk (ez az alapértelmezés)
DISTINCT az azonos sorok közül csak 1-et tartunk meg.
* minden oszlopot kiválasztunk a kiválasztásra.
oszlopnév csak a felsorolt oszlopokat választjuk ki.
kifejezés konstans vagy oszlopokon és/vagy konstansokon végzett megengedett aritmetikai vagy sztring műveletek eredménye. Pl. összeg*százalék/100
FROM táblázatnév: amelyik táblázatból kell kiválasztani a megadott oszlopokat.
WHERE kiválasztási feltétel: amely feltételnek megfelelő adatokra vagyunk kíváncsiak
GROUP BY oszlopnév a feltétel alapján kiválasztott sorok ezen oszlopok értékei szerint lesznek csoportosítva
HAVING csoportkiválasztási feltétel azon csoportok jelennek meg a SELECT végeredményében, melyekre az itt megadott kiválasztási feltétel teljesül.
ORDER BY oszlopnév a kiválogatott adatok a megadott oszlop szerint rendezve jelennek meg.

SQL oszlopfüggvények

Az argumentumban megadott oszlop, kifejezés, vagy SQL függvény értékét adják meg a kiválasztott sorokra. AVG(), MAX(), MIN(), SUM(), COUNT(*) sorok számát adja meg.

Adatváltoztató utasítások

Táblázatok tartalmát változtatják meg.

Törlés (DELETE utasítás)
Sort, vagy sorokat töröl ki egy táblázatból.
DELETE FROM táblázatnév
WHERE kiválasztási feltétel
Ha WHERE nélkül adjuk meg, akkor a összes sorát törli

Adatbevitel (INSERT utasítás)
Sort, vagy sorokat helyez be egy táblázatba.
Numerikus oszlopba csak numerikus, karakteres oszlopba csak karakteres érték vihető be. Ha a bemenő adat bármelyik mezője nem fér el a megfelelő oszlopban, akkor a sor nem kerül be a táblázatba.
INSERT INTO táblázatnév (oszlopnév1,...)
VALUES ([konstans, NULL],...)
A VALUES után megadott értékek sorrendje meg kell egyezzen a megadott oszlopsorrenddel.

Módosítás (UPDATE utasítás)
Módosítja a megadott táblázat egy vagy több sorának meghatározott oszlopát, vagy oszlopait.
UPDATE táblázatnév
SET oszlopnév1=[kifejezés, NULL],...
WHERE kiválasztási feltétel
SET a módosítani kívánt oszlopokat kell felsorolni utána.
A módosítani kívánt sorokat a WHERE feltétellel választjuk ki.

Vezérlő utasítások

Jogosultság megadása (GRANT utasítás)
A GRANT utasítás meghatározott műveletekre meghatározott felhasználóknak
- privilégiumokat, jogosultságokat ad meghatározott táblázatokon,
- engedélyezi SQL programok futtatását,
- engedélyezi az egész adatbázis állapotának megváltoztatását.

GRANT [ALL, ALTER, DELETE, INDEX, INSERT, SELECT, UPDATE, UPDATE (oszlopnév1,...)]
ON TABLE táblázatnév
TO [felhasználói azonosító, PUBLIC] [WITH GRANT OPTION]
ALL minden jog
ALTER tábla szerkezet módosítás
PUBLIC minden felhasználó számára
WITH GRANT OPTION az adott felhasználó a jogait továbbadhatja, ill. visszavonhatja.

Egyszerű lekérdezések

A lekérdezéseket gyakorlaton próbáljuk ki, itt csak SQL nyelven megfogalmazott kérések és magyarázatuk szerepel, nem pótolja az órán való részvételt!

```
SELECT * FROM alkalmazottak
```

Eredmény: minden adatok megjelenít az alkalmazottak táblából

A *relációs algebrai vetítésnek megfelelő művelet*, mellyel csak a számunkra szükséges adatok lekérdezését kérjük. A SELECT kulcsszó után soroljuk fel, mely attribútumok adatait szeretnénk látni a lekérdezésben. A relációs algebrában a projekció műveletnek felel meg:

```
SELECT részleg_az, név, főnök_az FROM alkalmazottak
```

Eredmény: csak a részlegazonosítót, a nevet és a főnökazonosítót mutatja meg.

```
SELECT név, fizetés * 12 FROM alkalmazottak
```

Eredmény: kiírja a nevet és az éves fizetés mértékét.

Alternatív oszlopnevek használata:

```
SELECT név, fizetés * 12  AS  éves_fizetes FROM alkalmazottak
```

Eredmény: az AS kulcsszó miatt nem a rendszer által adott oszlopnév szerepel az éves fizetési értékek felett, hanem az általunk megadott elnevezés.
Itt utalunk vissza az átnevezés relációs algebrai műveletre, mely itt mutatkozik meg. A lekérdezésben akár számított oszlopértéknek, akár létező mezőnek adhatunk a sémában megadottnál eltérő nevet a lekérdezés idejére. Így ékezettel láthatjuk el, vagy pl. magyarosíthatjuk az oszlopfejléceket a lekérdezésben.

Duplikált sorok kiküszöbölése:

```
SELECT DISTINCT részleg_az FROM alkalmazottak
```

Eredmény: az azonos értékek alapból annyiszor jelennek meg a lekérdezésben, ahányszor előfordulnak a relációban. Az ismételt kiírást lehet a DISTINCT kulcsszóval kikapcsolni.

Adatok rendezése:

```
SELECT részleg_az , név FROM alkalmazottak ORDER BY részleg_az
```

Eredmény: az adatok részlegazonosító szerint rendezetten jelennek meg a lekérdezésben.

```
SELECT részleg_az , név FROM alkalmazottak ORDER BY 2 DESC
```

Eredmény: az adatok név szerint (másodikként felsorolt mező a SELECT után) csökkenő sorrendben jelennek meg a lekérdezésben.

Válogatás a sorok között
(kiválasztás, szelekció a relációs algebrában)

```
SELECT *
FROM alkalmazottak
WHERE fizetés<120000;
WHERE belépés>1991.01.01.
WHERE jutalék>500000
WHERE fizetés<=245000
WHERE főnök_az IN (7902, 7566, 7788)
WHERE név LIKE 'S%'      /* S-sel kezdődő nevek*/
WHERE név LIKE '_____'   /*5 karakter hosszúságú nevek*/
WHERE főnök_az IS NULL   /* a főnök_az nincsen kitöltve */
WHERE név='Király'
```

A WHERE feltétellel fogalmazzuk meg, mely sorokat szeretnénk megjeleníteni a lekérdezésben. A relációs algebrában a szelekció műveletének felel meg.
A fenti WHERE feltételekből a lekérdezésben csak 1-1 adható meg, az egyszerűség kedvéért soroltuk csak fel egymás alá őket.

További példák LIKE-ra:

```
 LIKE 'a%'     minden 'a' betűvel kezdődő
 LIKE 'x_'     minden 'x'-el kezdődő kétbetűs
 LIKE '%a%'    minden 'a' betűt tartalmazó
 LIKE '_a%x'   második betű 'a' és 'x'-re végző
```

Összetett feltételek:

```
SELECT DISTINCT azonosító, név, foglalkozás, fizetés
FROM alkalmazottak
WHERE fizetés BETWEEN 100000 AND 200000
AND foglalkozás='hivatalnok'
```

Eredmény: Kiírja azok adatait, akiknek a fizetése 100.000 és 200.000 közötti **és** hivatalnokként dolgoznak a cégnél

```
SELECT DISTINCT azonosító, név, foglalkozás, fizetés
FROM alkalmazottak
WHERE fizetés BETWEEN 100000 AND 200000
OR foglalkozás='hivatalnok'
```

Eredmény: Kiírja azok adatait, akiknek a fizetése 100.000 és 200.000 közötti **vagy** hivatalnokként dolgoznak a cégnél

Az AND és OR logikai művelet használatával tudunk megfelelő összetettségű feltételt megadni a lekérdezéshez.

Csoportfüggvények
Sorok egy csoportján hajtódnak végre, és minden csoportra egy-egy eredményt adnak vissza.

A függvények: AVG, COUNT, MAX, MIN, SUM, (VARIANCE, STDDEV)

```
SELECT MIN(fizetés), MAX(fizetés), SUM(fizetés), AVG(fizetés),
COUNT(fizetés) FROM alkalmazottak
```

Eredmény: kiírja az alkalmazottaknál a legalacsonyabb, legmagasabb fizetés értékét, az összegüket, az átlagukat és azt, hányan kapnak fizetést.
Megjegyzés: Csoportfüggvény használatánál a SELECT után más attribútum nem szerepelhet, kivéve a GROUP BY-nál használt mező nevét

A csoportfüggvények jelentése:

AVG (attribútum)	átlag
COUNT (attribútum)	nem NULL elemek száma
COUNT (*)	sorok száma NULL is
MAX (attribútum)	maximális elem
MIN (attribútum)	minimális elem
SUM (attribútum)	összeg
STDDEV(attribútum)	szórás

```
SELECT MIN(fizetés) AS legkisebb_fiz, MAX(fizetés) AS
legnagyobb_fiz, SUM(fizetés) AS összeg, AVG(fizetés) AS átlag ,
COUNT(fizetés) AS darab FROM alkalmazottak
```

Eredmény: azonos a fenti példával, azzal a különbséggel, hogy mi adtunk az oszlopoknak egyéni nevet.

Csoportosítás:

```
SELECT részleg_az, MIN(fizetés) AS legkisebb_fiz, MAX(fizetés)
AS legnagyobb_fiz, SUM(fizetés) AS összeg, AVG(fizetés) AS
átlag , COUNT(fizetés) AS darab FROM alkalmazottak
GROUP BY részleg_az
```

Eredmény: a GROUP BY kulcsszó az utána megadott oszlop szerinti azonos értékű sorokra külön csoportokat készít. Kiírja a részlegazonosítót és a fizetések legkisebb, legnagyobb értékét, összegüket, átlagukat és azt, hányan kapnak fizetést az egyes részlegekre nézve.
Jó tanács: Amikor úgy fogalmazzunk meg a feltételt magyarul, hogy ...-ként, pl. részlegenként, akkor GROUP BY-t kell használni, arra az oszlopra, amely nevénél használtuk a -ként toldalékot.

```
SELECT részleg_az, AVG(fizetés) AS átlag
FROM alkalmazottak GROUP BY  részleg_az
HAVING COUNT(*)>3
```

Eredmény: Csak azon részlegekre csoportosított adatokat látjuk, ahol a dolgozók létszáma nagyobb, mint 3. A HAVING a már meglévő csoportokat szűri, NEM használható az egész tábla szűrésére, arra a WHERE használandó.

Lekérdezés több táblából:

join = természetes összekapcsolás I

```
SELECT név, foglalkozás, részleg_név FROM alkalmazottak,
részleg
WHERE alkalmazottak.részleg_az = részleg.részleg_az
```

Eredmény: megmutatja a név és foglalkozás mellett, hogy ki, melyik elnevezésű részlegen dolgozik. Ehhez össze kell kapcsolni a két táblát a közös attribútum a részlegazonosító alapján. Mivel nem lenne egyértelmű a fordító számára, hogy melyik részlegazonosítóra is hivatkozunk, meg kell adnunk előtte ponttal elválasztva a reláció nevét is.

join = természetes összekapcsolás II

```
SELECT név, foglalkozás, részleg_név
FROM alkalmazottak
INNER JOIN részleg ON alkalmazottak.részleg_az =
részleg.részleg_az
```

Eredmény: ugyanaz. mint az előző lekérdezésben, de már nem WHERE feltétellel megoldva az összekapcsolást. Az alap SQL nyelv az elő megoldást támogatta sokáig, amíg meg nem jelent a LEFT és a RIGHT JOIN lehetőség az INNER JOIN mellett.
Használata:

```
SELECT [reláció.]attribútum, [reláció.]attribútum, ...
FROM első_tábla INNER JOIN második tábla
ON első_tábla.kulcs_mező = második_tábla.kulcs_mező
```

Az INNER JOIN esetében amelyik sorhoz a kapcsolómező értéke alapján nem tartozik érték a másik táblából, azt nem írja ki, amelyikhez több tartozik, azt annyiszor, ahány sor a másik táblában hozzá kapcsolható (ezért azonosan működik a WHERE feltételes megoldással).
Az INNER általában elhagyható, vagy írható helyette LEFT, RIGHT, FULL OUTER, CROSS.
LEFT esetén: az első tábla adatai akkor is szerepelnek, ha nincs illeszkedő adat a másikban
RIGHT esetén: az másik tábla adatai akkor is szerepelnek, ha nincs illeszkedő adat az elsőben
FULL OUTER: mindkét táblából megmutatja az összes sort
CROSS: a táblák Descartes szorzatát képezi, azaz az összes lehetséges kombinációt megmutatja:
SELECT * FROM első tábla CROSS JOIN második tábla

Egymásba ágyazott lekérdezések:
Előfordul, hogy lekérdezéseket kell egymásba ágyazni, mert a belső lekérdezés eredményével tud a külső lekérdezés dolgozni. Ilyenkor először a belső lekérdezés fut le/értékelődik ki, majd annak eredményét megkapja a külső lekérdezést, hogy ő is elindulhasson. Két eset lehet: A belső lekérdezés egy értéket, vagy a belső lekérdezés több értéket ad vissza eredményként. Mindkét esetre nézünk példát.

A belső SELECT egy értéket/sort ad:
Keressük meg a legkisebb fizetésű dolgozót!

```
SELECT név, fizetés, részleg_az
FROM alkalmazottak
WHERE fizetés= (SELECT  MIN(fizetés)
                FROM  alkalmazottak)
```

Eredmény: a belső lekérdezés visszaadja az alkalmazottak közül a legalacsonyabb fizetés értékét és ezt az értéket használja fel a külső lekérdezés a WHERE feltételben.

A belső SELECT több sort ad vissza:
Keressük ki azokat a dolgozókat, akik többet keresnek, mint a legkisebb fizetés a 30-as osztályon!

```
SELECT fizetés, foglalkozás, név, részleg_az
FROM alkalmazottak
WHERE fizetés> SOME ( SELECT fizetés
                      FROM alkalmazottak
                      WHERE részleg_az=30)
```

Eredmény: a belső lekérdezés visszaadja az alkalmazottak közül a 30-as részlegen dolgozók fizetéseit, majd a külső lekérdezés a WHERE feltételében megnézi, kik azok, akik valamelyik (SOME) visszaadott fizetésnél jobban keresnek, tehát elég, ha a legalacsonyabb visszaadott értéknél többet keresnek.

Keressük ki azokat a dolgozókat, akik többet keresnek, mint BÁRKI a 30-as osztályon!

```
SELECT fizetés, foglalkozás, név, részleg_az
FROM alkalmazottak
WHERE fizetés>ALL(SELECT fizetés
                  FROM alkalmazottak
                  WHERE részleg_az=30)
```

Eredmény: a belső lekérdezés visszaadja az alkalmazottak közül a 30-as részlegen dolgozók fizetéseit, majd a külső lekérdezés a WHERE feltételében megnézi, kik azok, akik mindegyik (ALL) visszaadott fizetésnél jobban keresnek, tehát a legmagasabb visszaadott értéknél is többet keresnek.

Keressük meg minden osztályon a legkisebb fizetésű dolgozót!

```
SELECT név, fizetés, részleg_az
FROM alkalmazottak
WHERE  (fizetés, részleg_az) IN
                        (SELECT  MIN(fizetés),
                        részleg_az FROM alkalmazottak
                        GROUP BY részleg_az);
```

Eredmény: a belső lekérdezés visszaadja az alkalmazottak táblából a három részleg azonosítóját és az adott részlegben legalacsonyabb fizetést, majd a külső lekérdezés a WHERE feltételében megnézi, kik azok, akiknél a részlegazonosító és az adott fizetési érték azonos, az ő adatait jeleníti meg.

Halmazműveletek SQL-ben

Válasszuk ki azokat a termeket, ahol a GI1-nek vagy a GI2-nek vannak órái:

```
SELECT Tanterem FROM etr WHERE Osztaly = 'GI1'
UNION
SELECT Tanterem FROM etr WHERE Osztaly = 'GI2';
```

Azon évfolyamok, melyeknek a Zsigmond 1 és a Deák teremben is van órája:

```
SELECT Evfolyam FROM etr WHERE Tanterem = 'Zsigmond 1'
INTERSECT
SELECT Evfolyam FROM etr WHERE Tanterem = 'Deák';
```

Azon évfolyamok, melyeknek a Zsigmond 1-es teremben van, de a Deák teremben nincs órája:

```
SELECT Evfolyam FROM etr WHERE Tanterem = 'Zsigmond 1'
MINUS
SELECT Evfolyam FROM etr WHERE Tanterem = 'Deák';
```

Gyakorló feladatok SQL nyelven I.

Hozzunk létre egy új könyvtári adatbázist a következő szerkezetű táblákkal:

Könyvek
könyvkód (INTEGER)
Szerző (VARCHAR(25))
Cím (VARCHAR(25))

Olvasók:
olvasókód (INTEGER)
név (VARCHAR(25))
irszám (DEC(4))
város (VARCHAR(25))
utca (VARCHAR(30))

Kölcsönzés
könyvkód (INTEGER)
olvasókód (INTEGER)
dátum (DATE)

Megjegyzés:
Az adatbázis, tábla és mezőnevek megadásánál nem használunk ékezetet, hogy a kompatibilitás az egyes rendszerek között ne kockáztassuk ezzel.

A táblák létrehozása a CREATE TABLE paranccsal történik.
A könyvek tábla létrehozása:

```
CREATE TABLE konyvek (
konyvkod INTEGER NOT NULL,
szerzo VARCHAR(25) NOT NULL,
cim VARCHAR(25) NOT NULL,
PRIMARY KEY (konyvkod)); (elsődleges kulcs létrehozása)
```

Olvasók tábla létrehozása:

```
CREATE TABLE olvasok (
olvasokod INTEGER NOT NULL,
nev VARCHAR(25) NOT NULL,
irszam DEC(4) NOT NULL,
varos VARCHAR(25) NOT NULL,
utca VARCHAR(30) NOT NULL,
PRIMARY KEY (olvasokod, nev));
```

A kölcsönzés tábla létrehozása:

```
CREATE TABLE kolcsonzes (
konyvkod INTEGER NOT NULL,
olvasokod INTEGER NOT NULL,
datum DATE DEFAULT "NOW" NOT NULL,
PRIMARY KEY (konyvkod, olvasokod, datum));
```

Megjegyzés:
A dátum mező megadásánál nem biztos, hogy minden rendszer engedi az alapértelmezett érték beállítását, ezt érdemes ellenőrizni a tábla létrehozása előtt.

Vigyünk fel adatokat a táblákba.

```
INSERT INTO konyvek VALUES (1,"Robert Merle", "Malevil")

INSERT INTO olvasok (olvasokod, varos, utca, irszam, nev)
VALUES (1, "Budapest", "Sas u. 7.", 1051, "Magyar Sándor")
```

(saját sorrendben visszük fel az adatokat, ezért kellet felsorolni külön a mezőneveket is a megfelelő sorrendben)

```
INSERT INTO kolcsonzes (konyvkod, olvasokod) VALUES (1,1)
```

(a dátumot a rendszer adja, amennyiben működik a DEFALUT értékmegadás az adott rendszerben)

Nézzük meg a felvitt adatokat:

```
SELECT * from táblanév
```

Vigyünk fel minél több adatot a táblákban, hogy tudjunk miből lekérdezést készíteni!

Nézzük meg a felvitt adatokat lekérdezés eredményeként:

Kik milyen könyveket kölcsönöztek ki?

```
select nev, szerzo, cim from olvasok, konyvek, kolcsonzes
where (kolcsonzes.olvasokod=olvasok.olvasokod) and
(kolcsonzes.konyvkod=konyvek.konyvkod)
```

1, Adott nevű olvasó milyen könyveket kölcsönzött?

```
select szerzo, cim from olvasok, konyvek, kolcsonzes where
(kolcsonzes.olvasokod=olvasok.olvasokod) and
(kolcsonzes.konyvkod=konyvek.konyvkod) and
(olvasok.nev='Magyar Sándor')
```

2, Adott szerző műveit kik olvassák?

```
select nev from olvasok, konyvek, kolcsonzes where
(kolcsonzes.olvasokod=olvasok.olvasokod) and
(kolcsonzes.konyvkod=konyvek.konyvkod) and
(konyvek.szerzo='William Shakespeare')
```

3, Hány könyve van a könytárnak?

```
select count(*) from konyvek
```

4, Hány olvasója van a könytárnak?

```
select count(*) from olvasok
```

5, Hány db könyv van kölcsönözve jelenleg?

```
select count(*) from kolcsonzes
```

6, Mely olvasók budapestiek?

```
Select nev, varos, irszam, utca from olvasok where
varos='Budapest'
```

7, Kik azok az olvasók, akik április 12 után kölcsönöztek ki könyvet? (egy név csak 1-szer szerepeljen)

```
select distinct nev from olvasok, konyvek, kolcsonzes where
(kolcsonzes.olvasokod=olvasok.olvasokod) and
(kolcsonzes.datum>"02-12-2021")
```

8, Kik azok az olvasók, akik február 12-én kölcsönöztek ki könyvet? (egy név csak 1-szer szerepeljen)

```
select distinct nev from olvasok, konyvek, kolcsonzes where
(kolcsonzes.olvasokod=olvasok.olvasokod) and
(kolcsonzes.datum>"02-11-2021") and (kolcsonzes.datum<"02-13-
2021")

select distinct nev from olvasok, konyvek, kolcsonzes where
(kolcsonzes.olvasokod=olvasok.olvasokod) and (kolcsonzes.datum
between ("02-11-2021") and ("02-13-2021"))
```

9, Mely olvasók neve kezdõdik R betűvel?

```
select nev from olvasok where nev like "R%"
```

10, Városonként csoportosítva hány ember olvasója a könyvtárnak?

```
select varos, count(*) from olvasok group by varos
```

Gyakorló feladatok SQL nyelven II.

Hozzuk létre a következő szerkezetű táblákat:

Jegy
nev= diak neve (VARCHAR(30))
mat= matek jegy (NUMERIC(1))
fiz= fizika jegy (NUMERIC(1))
kem= kemia jegy (NUMERIC(1))
szak= melyik szakra jar (VARCHAR(5))

Csoport
szak= melyik szakra jar (VARCHAR(5))
patron= patronáló tanár(VARCHAR(30))

A következő lekérdezéseket kellene kipróbálni:

1, diákok neve, szakja, partonáló tanárának neve:

```
select a.nev, a.szak, b.patron
from jegy a, csoport b
where a.szak=b.szak
```

2, X Y csoporttársai

```
select b.nev, b.szak
from jegy a, jegy b
where a.nev="X Y" and a.szak=b.szak and a.nev<>b.nev
```

3, X Y tanár csoportjába tartozó diákok

```
select nev, szak
from jegy
where szak=(select szak from csoport where csoport.patron=”X
Y”)
```

4, X Y ill. M P tanár csoportjába tartozó diákok

```
select nev, szak
from jegy
where szak=(select szak from csoport where csoport.patron=”X
Y”)
or szak=(select szak from csoport where csoport.patron=”M P”)
```

Másik megoldás UNION-al

```
select nev, szak
from jegy
where szak=(select szak from csoport where csoport.patron=”X
Y”)
union
select nev, szak
from jegy
where szak=(select szak from csoport where csoport.patron=”M
P”)
```

5, Azon diákok neve, akik matekból az átlagosnál jobban szerepeltek

```
select nev, mat
from jegy
where mat>(select avg(mat) from jegy)
```

6, Azon diákok neve, akik matekból és fizikából is az átlagosnál jobban szerepeltek

```
select nev, mat, fiz
from jegy
where mat>(select avg(mat) from jegy)
and fiz>(select avg(fiz) from jegy)
```

7, Szakonként a mat, fiz ill. kem jegyek átlagainak átlaga

```
select avg(mat+fiz+kem)/3,szak
from jegy
group by szak
order by 1 descending
```

8, Csoportonként a mat, fiz ill. kem jegyek átlagainak átlaga, ahol a szakátlag jobb, mint 3.6

```
select avg(mat+fiz+kem)/3,szak
from jegy
group by szak
having avg(mat+fiz+kem)/3>3.6
order by 1 descending
```

9, Azon diákok nevei, akiknél a három jegy átlaga jobb, mint a leggyengébb szak átlaga

```
select nev, (mat+fiz+kem)/3
from jegy
where (mat+fiz+kem)/3> any (select avg((mat+fiz+kem)/3) from
jegy group by szak)
```

10, Azon diákok nevei, akiknél a három jegy átlaga jobb, mint a legjobb szak átlaga

```
select nev, (mat+fiz+kem)/3
```

```
from jegy
where (mat+fiz+kem)/3> all (select avg((mat+fiz+kem)/3) from
jegy group by szak)
```

11, Azon diákok nevei, akiknél a három jegy átlaga jobb, mint a saját szakán elért átlag

```
select a.nev, (a.mat+a.fiz+a.kem)/3
from jegy a
where (a.mat+a.fiz+a.kem)/3> (select avg((b.mat+b.fiz+b.kem)/3)
from jegy b where b.szak=a.szak)
```

X. Irodalomjegyzék

[1] George M. Marakas - Decision Support Systems In The 21st Century, Publisher: Pearson College Div, 2002, ISBN: 978-8120323766

[2] Quittner Pál - Adatbázis-kezelés a gyakorlatban, Akadémiai Kiadó, 1993, ISBN: 0729000912914

[3] Codd, Edgar F. (June 1970). "A Relational Model of Data for Large Shared Data Banks". Communications of the ACM. 13 (6): 377–87.

[4] Hector Garcia-Molina and Jeffrey D. Ullman and Jennifer Widom - Database Systems: The Complete Book, Publisher: Prentice Hall, 2002, ISBN: 978-0131873254

[5] A.P.G. Brown, "Modelling a Real-World System and Designing a Schema to Represent It", in Douque and Nijssen (eds.), Data Base Description, North-Holland, 1975, ISBN 0-7204-2833-5.

[6] Chatham, Mark (2012). Structured Query Language By Example - Volume I: Data Query Language. p. 8. ISBN 978-1-29119951-2.

Kiadó – Vydavateľ:
Selye János Egyetem – Univerzita J. Selyeho,
Bratislavská cesta 3322,
SK-945 01 Komárno
www.ujs.sk

Cím - Názov:
ADATBÁZIS-KEZELÉSI ALAPISMERETEK

Szerzők – Autori:
Dr. Kiss Gábor

Lektorok - Recenzenti:
Ing. Ondrej Takáč, PhD.
RNDr. Árki Zuzana, PhD.

Terjedelem – Rozsah:
4,2 szerzői ív – 4,2 AH

Kiadás éve - Rok vydania:
2021

ISBN 978-80-8122-389-1

ISBN 978-80-8122-389-1