SSADM
Structured Systems Analysis and Design Method
(strukturált rendszerelemzési és tervezési módszertan)

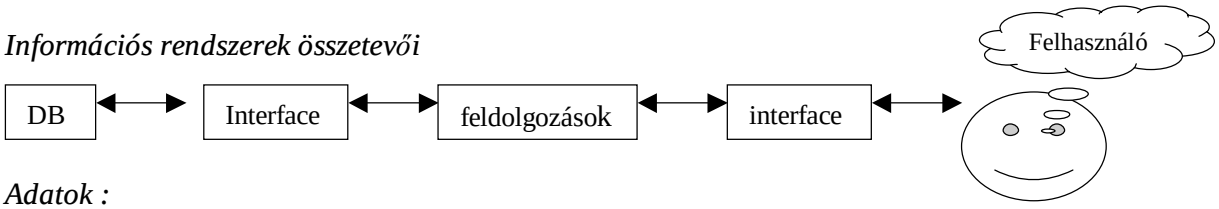
Történelmi áttekintés:

1. Feldolgozásközpontú (folyamatorinetált):
   - folyamatorientált,
   - feldolgozási egységekre koncentráló,
   - output-célú szemlélet,
   - úgynevezett hagyományos tervezési módszer
2. Adatközpontúság:
   - középpontban az adatok
   - cél: redundancia megszüntetése
   - cél: adatok egységesítése
   - középpontba került az adatmodellezés
3. Struktúrált módszertanok
   - sem a folyamatoknak sem az outputoknak, sem az adatoknak nincs elsőbbsége
   - programokat sokkal könnyebben el lehet készíteni, ha modulokra bontjuk őket, és a modulokat összekapcsolva egy szerkezetet, struktúrát alakítunk ki.
   - a struktúrálásra szabályokat lehet kidolgozni, ezen alapul a struktúrált programozás elmélete és gyakorlata is.

Összegzés : Az SSADM olyan struktúrált rendszertervezési módszertan, amely a fejlesztés elemzési és tervezési fázisait támogatja, és eleget tesz a struktúrált módszertanokkal szemben támasztható valamennyi követelménynek. Felépítésében három nagyobb részt tartalmaz. Strukturális része az elvégzendő tevékenységek időbeliségével foglalkozik, technikai része azt mondja meg, hogyan kell a tevékenységeket elvégezni, adatszótára pedig leírja az előállítandó termékeket.

Bevezető alapok:

Információs rendszerek összetevői



Adatok :

- bementi (input)
- kimeneti (output)
- tárolt adatok (adatbázis)

Folyamatok :

- lekérdező
- karbantartó

A rendszerszervezési munka menete :

- mi az előállítandó eredmény
- milyen részekre bontható fel a munka

- milyen típusú munkaerőt igényel a feladat

A problémamegoldás általános sémája :

1. A feladat felismerése és megfogalmazása
2. A helyzet elemzése, megoldási vázlatok átgondolása
3. A megoldás tervezése
4. A szükséges eszközök kivitelezése
5. Megoldás az elkészített eszközök használatával

SSADM SZERKEZETE:

Három dimenziója van :

- Adatok
- Folyamatok
- Idő

Szerkezete :

- Hierarchikus
- Felülről-lefele haladó (up-down)
- Modulok -> szakaszok -> lépések -> feladatokra van osztva

A technikák feladatokhoz kapcsolódnak.

SSADM TECHNIKÁI ÁLTALÁNOSAN:

Az SSADM kb. egy tucatnyi technikát foglal magába. Azért kb., mert annak olyan módszerek, amelyeket nem minden szakember sorol az önálló technika kategóriájába.

Adatokhoz kapcsolódó :

- Logikai adatmodellezés
- Relációs adatmodellezés
- Fizikai adattervezés

Folyamatokhoz kapcsolódó :

- Adatfolyam-modellezés
- Funkció-meghatározás
- I/O tervezés
- Dialógustervezés
- Menütervezés
- Logikai adatbázis folyamatok tervezése
- Fizikai folyamattervezés

Idővel kapcsolatos

- Egyedtörténeti diagrammok
- Esemény-hatás diagrammok
- Logikai adatbázis folyamatok tervezése

– Fizikai folyamat és adattervezés

A technikák két fő csoportba oszthatók:
1. diagramra épülők
2. nem diagramszerűek

Mivel az SSADM alapvetően a rendszerfejlesztéshez szükséges kommunikációt kell, hogy támogassa, sok esetben használja ki a képi ábrázolás lehetőségét.

Diagramszerű technikák:
- logikai adatmodellezés
- adatfolyam modellezés
- egyed-esemény modellezés
- elérési út modellezése
- I/O szerkezet meghatározása
- dialógustervezés
- feldolgozások logikai modellezése

Azért mondjuk, hogy ezek diagramra épülnek, mert a diagramokon kívül még sok információ van, ami a diagramon nem adható meg. Ezek lehetnek kézzel készültek, ill. számítógéppel készültek. A kézi dokumentációhoz formanyomtatványokra van szükségünk.

Nem diagramszerű technikák:
- követelmény meghatározás
- rendszertervezési változatok kidolgozási módszere
- rendszertechnikai változatok kidolgozási módszere
- funkció meghatározás
- relációs adatelemzés
- specifikáció prototipizálás
- fizikai tervezés

Folyamatmodellezés:

(ennek segítségével tudja leírni a rendszerszervező az információs rendszereket)
Célja :
- az információs rendszerekben zajló folyamatok
- az adatok tárolásának, mozgásának valamint forrásainak és felhasználásának leírása

A leírással szemben támasztott követelmények :

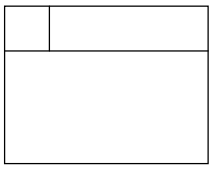
- könnyen és gyorsan előállítható
- könnyen és gyorsan változtatható
- könnyen érthető mind a szakemberek, mind a számítástechnikában nem járatos emberek számára
- tegyen lehetővé megfelelő részletezettséget

Az eszköz : adatfolyam-diagram

- gyakorlatiasak, nem technikai jellegű -> gyorsan elkészíthető és módosítható
- könnyen érthető a felhasználó számára -> ezért támogatja a fejlesztő és a felhasználó közti kommunikációt
- szintekre bontja -> támogatja a felülről lefelé haladó szemléletet
- minimális átfedés -> könnyű a változások kezelése

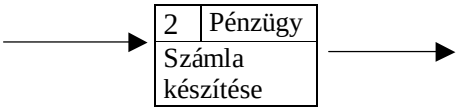
Adatfolyam diagram elemei :

- **Folyamat**



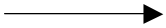
Eljárásokat jelölnek, amelyek a rendszer azon tevékenységei, amelyek az információ-átalakítást végzik. Téglalappal jelöljük, amelynek bal felső sarkában van az azonosító: ez egy szám. Jobb felső sarkában a szervezeti egység neve van, ami a végrehajtásért felelős. Az alsó részben a folyamat megnevezése található, egy tevékenységet kell, hogy jelöljön.

Példa:



2 : azonosító (nem jelent sorrendiséget)
Pénzügy : szervezeti egység neve, ő felelős az adott folyamat végrehajtásáért
Számla készítése : eljárás (átalakítja, illetve kezeli az adatokat)

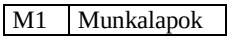
- **Adatfolyam**

Ezen keresztül áramlanak az adatok a diagram elemei között. 

- **Adattár**

Jobbról nyitott, keskeny téglalap jelképezi az adattárakat.
Lehet ez manuális adattár (kartotékfiók, file).
A gyakran használt adatokat törzsadattárakban tároljuk.
Ha olyan adataink vannak, amelyekre nem azonnal van szükségünk, átmeneti adattárakba kerülnek. Az adattárakat azonosítóval látjuk el. Ezeket a bal oldali kis négyzetbe kell írni. Betűből és számból állnak. M betű jelöli a manuális adattárat, D a számítógépeset. Az ú.n. logikai AFD-ken L betűt is használunk. Fontos, hogy ugyanaz az adattár több helyen is megjelenhet az AFD-n, ilyenkor ezt jelölni kell: a bal oldali határoló vonalat megkettőzzük.

Példa: Adattár



M1 : azonosító

Munkalapok : megnevezés

D1	Számítógépes főadattár

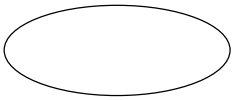
D2	Ismétlődő számítógépes adattár

M1	manuális adattár

T1	Átmeneti adattár

1. Környezeti elem

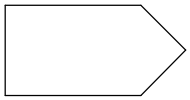
   A rendszer határain kívül eső adatforrást, vagy adatok címzettjét jelöli. Azért lényeges, hogy ezeket ábrázoljuk, mert ezek az elemek nem részei a rendszerünknek, de a működését befolyásolják. Szimbóluma egy ellipszis alak.



   Ilyen elemek közti közvetlen adatfolyamokat nem szokás jelölni, ha azonban szükséges, szaggatott vonallal kell ábrázolni.
   Ezt az elemet is fel lehet tüntetni az ADF-n több helyen, ilyenkor egy ferde vonalat húzunk az ellipszisbe.

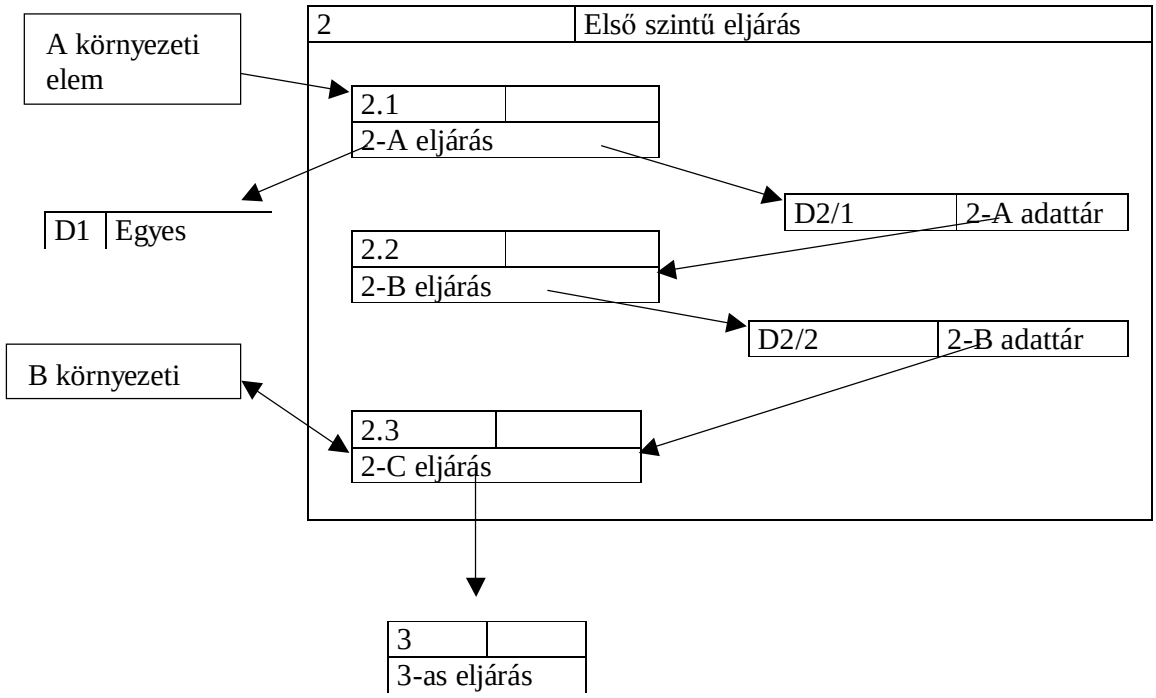
2. Anyagfolyam

   Anyagok fizikai áramlását csak ritkán kell jelölni, ha segíti a könnyebb megértést.



Az AFD-k elemei – mint láttuk – kapcsolatban állnak egymással, azonban nem létesíthetünk kapcsolatot bármely kettő elem között nyilak alkalmazásával.
Pl.: adattárral nem állhat közvetlen kapcsolatban egy környezeti elem, csak egy folyamaton keresztül kerülhetnek kapcsolatba egymással.

„De Marco szabály” : Amikor egy rendszer túl nagy ahhoz, hogy egyetlen lapon ábrázolhassuk, akkor folyamatait részfolyamatokra, valamint ezek közötti adatfolyamokra bontjuk fel. Ezt a felbontást akár több hierarchiaszinten át folytathatjuk. Végeredményként AFD-k szintekre bontotot halmazát kapjuk.



- A környezet, a folyamatok, az adattárak és az adatfolyamok alkotják a felülről lefele haladó rendszerleírás felső csúcsát. Mivel a diagramok méretét az áttekinthetőség és tárolhatóság kedvéért A4 formátumra korlátozták, és egy lapon legfeljebb 7 folyamatot illik csak ábrázolni, a „folyamat” elnevezés nem feltétlenül fedi a valóságot. Valójában alrendszernek, fő funkciónak vagy eljáráscsoportnak kellene hívni, legalábbis ezen a legfelső szinten.
- Az adattárak közül is csak azokat tüntetjük itt fel, amiket legalább 2 ábrázolt funkció használ.A részletekhez úgy juthatunk el, ha ezen szint funkcióit külön-külön ábrázoljuk újabb, alacsonyabb szintű AFD-ken.
- Előfordul, hogy az alacsonyabb szintű AFD készítésekor kiderül, hogy egy felsőbb szinten valamit nem, vagy nem helyesen ábrázoltunk. Ilyenkor azokat természetesen módosítani kell, hiszen az egyes szintek diagramjai között ellentmondás nem lehet. Hogy tényleg ne is legyen, be kell tartanunk egy-két szabályt.

1. Magasabb szintű AFD minden egyes eljárása külön-külön fejtendő ki egy-egy alacsonyabb szintű diagramon.
2. Az alacsonyabb szintű AFD-n szereplő eljárásokat a magasabb szintű eljárás azonosítójának nevéből származtatjuk. Pl. ha a felső szintű eljárásokból a 4-est részletezzük, akkor az ezen szinten lévő eljárások azonosítói a 4.1, 4.2…stb. lesznek.
3. A részletező szintű AFD-k készítéséhez olyan nyomtatványt használunk, amelyen egy keret látható. Ebbe rajzoljuk a diagramot, a kereten kívülre pedig vagy felsőbb szintű elemek (pl.: adattár), vagy környezeti elem ábrázolható.
4. Ha vannak olyanok, melyek csak a részletező diagramon jelennek meg, akkor ezeket a kereten belül ábrázoljuk, azonosításkor pedig a betűjelük után először a magasabb szintű eljárásnak a számát írjuk, amit részletezünk, aztán egy „/” jel után a saját szinten belüli azonosítója következik.

5. Az olyan adattárak, amik már egy magasabb szinten megjelentek, de itt is ábrázolni akarjuk, a kereten kívülre kerülnek. Ha ugyanilyen szinten megjelenő adattárat szeretnénk ábrázolni, akkor azt is a kereten kívülre tehetjük.
6. Ha környezeti elemet kell ábrázolni, az a kereten kívülre kerül még akkor is, ha ezen a részletességi szinten jelenik meg először.
7. A részletező diagram belépő, ill. kilépő adatfolyamainak meg kell egyezniük a magasabb szintű diagram eljárásának ki-, ill. belépő adatfolyamaival. Ezt hívjuk egyensúly-szabálynak.
8. Részletező diagramon az egyes elemek szükség szerint felbonthatók a magasabb szintű diagramhoz képest.

A szükséges szintek száma:
Erre vonatkozó szabály nincs de a tapasztalat azt mutatja, hogy a három szintnél több szintet igénylő rendszerleírás ritka, sőt legtöbbször két szint is elegendőnek bizonyul.

Egy diagramon ábrázolandó részletek mennyiségének megadásakor az alábbiakat érdemes figyelembe venni:
- Figyeljünk arra, hogy a túl sok részlet megadásával ne tereljük el a figyelmet a fő folyamatokról.
- Úgy bontsuk szét a folyamatokat, hogy az így kapott részek között jól értelmezhető kapcsolat legyen.
- Számítsunk arra, hogy a mélyebb szintek felé haladva egy-egy diagramon a folyamatok száma a felette lévőhöz képest csökken.
- Legfontosabb: az a diagram a jó diagram, ami könnyen áttekinthető.

Az első, legfelső szintű (más néven áttekintő) diagram felett létezhet (nem kötelező) egy másik diagram, a kapcsolatdiagram. Ezen a szinten az egész rendszer egyetlen folyamat-blokkba sűrűsödik, amelyet a környezeti elemek (a rendszer környezete) vesznek körül.

Logikai adatmodellezés:

Folyamatmodellezés mellett a másik nagyon fontos
Az adatok feldolgozástól független, belső logikájára van szükség
Sokféle koncepció volt, de ebből kettő kiemelkedett az idők folyamán:
- Egyed-tulajdonság-kapcsolat modellezés (ETK) (by Bachman)
- Relációs modellezés (by Codd)

Célja: Valamely szervezet (vállalat, intézmény) működéséhez szükséges adatok feldolgozási követelményektől független és feldolgozó, ill. tároló eszközök lehetőségei által nem korlátozott leírása, belső, logikai szerkezetének meghatározása.

Ehhez kapcsolódik még másik két cél:
- Segítsen az alkalmazási terület minél jobb megértésében a fejlesztés valamennyi résztvevője (felhasználók, szervezők, programozók és egyéb szakemberek) számára
- Szolgáljon alapul az adattárolás konkrét megoldásának megtervezéséhez

A logikai adatmodell elemei:

- **Egyed**

Abból indul ki, hogy a rendszer, szervezet működéséhez a szervezetet alkotó embereknek egy sereg dologról tudniuk kell -> objektumok, egyedek.
Egyednek tekinthetünk bármit – konkrét, vagy akár elvont tárgyat, ill. fogalmat – amiről tudnunk kell vmit, vagy vmiket.
Az az egyed, aminek egy, vagy több tulajdonságát a szervezetben ismernie kell vkinek. Általánosításra törekszünk (Szabó János és SZEMÉLY obj. példája)

Egyednek van típusa (SZEMÉLY) és előfordulása (Janika)
Modellben egyedtípusokat ábrázolunk, méghozzá lekerekített sarkú téglalapban a névvel
pl. számla

- **Kapcsolat**
  A való világ objektumai, tehát az egyedek közti kapcsolat, viszony
  Kapcsolat állhat fenn két egyedtípus között, vagy ugyanazon egyedtípus két előfordulása között (egyedtípus önmagával létesített kapcsolata).
  Ennek is van típusa és előfordulása is.
  Itt is típust ábrázolunk mindig.
  - Jele: vonal (ügyfél - számla)
  - Kapcsolatnak van **foka** és **jellege**
  - ***Fok:*** az előfordulások szintjén milyenek a mennyiségi viszonyok a kapcsolódó egyedek között. Magyarul egyik típusúból mennyi kapcsolódhat a másik típusúhoz. Ha egyhez több is kapcsolódhat, akkor itt csak azt tudjuk, hogy sok, számot nem.
    Ezek szerint a kapcsolat foka lehet:
    Egyhez egy - 1:1
    Egyhez több - 1:n
    Többhöz több - n:m
    Több jele: seprű (a „több” oldalon)
    Az 1:n kapcsolat alárendelt és fölérendelt egyedtípusokat határoz meg.
  - ***Jelleg****:* mi az összefüggés az egyed-előfordulás léte és a kapcsolat léte között. (létezhet-e az egyed anélkül, hogy létezne az adott kapcsolatban hozzá kapcsolt másik egyed.)
    Jelölése: kötelező kapcsolatot folytonos vonallal, esetlegeset szaggatottal.
    A szaggatott azon az oldalon van, amelyik egyednek nem létkérdés a másik létezése.
    Pl: vevő – szaggatott – folytonos – seprű – megrendelés

Modellezési szempontok miatt minden kapcsolatot át kell alakítani 1:n típusúra.
Esetei:
1:n-t nem alakítom át, mer az jó nekem
1:1 az könnyű. Egyszerűen meggondolható, hogy az ilyet simán összevonhatom eggyé.
n:m – ez se nehéz, mert okos bácsik kitalálták, hogy mechanikusan hogyan csináljuk. A kapcsolatot mint új egyedet hozzuk létre, és az felé megy két seprű a két eredeti egyedtől. Az új egyed részéről a kapcsolat kötelező mindkét irányba (legyen mit összekapcsolni).

Speci eset: ugyanazon egyedtípus előfordulásai állnak egymással kapcsolatban: rekurzívnak hívjuk az ilyet.
Pl.: alkatrészek
Feloldása: az n:m-hez hasonlóan új egyedet vezetünk be…

- Két egyedtípus között meghatározható kapcsolatok száma nem korlátozott (bár ezek nem túl gyakoriak, és ritkán haladja meg számuk a négyet)
- Egy egyedtípus korlátlan számú kapcsolatban vehet részt
- Kapcsolatok nem ágaztathatók el és nem is találkoznak

Kapcsolattípusok egymás közti kapcsolatát is lehet vizsgálni. Pl.: Kizáró kapcsolatok
Ezen belüli pl:

Ezred tfh vagy csak Harckocsiból vagy csak Lövegből áll. (46. ábra p99) – ez kizáró alárendelt eset

Alkatrészt vagy gyárból vagy üzletből szerzünk be – Ez meg fölérendelő kizáró eset

Elemek azonosítása:

Egyedet a nevével (láttuk) – a neveknek egy adatmodellen belül egyedinek kell lenniük

Kapcsolatnak két nevet is adunk az eredeti angol nyelvű leírásnak megfelelően. Így állítólag pontosabban leírható a tényleges valóságbeli kapcsolat. Meg értelmes angol mondatot is könnyebb így az ábráról leolvasni.
Pl.: Order – containing – part of – Item -> Each ORDER must be CONTAINING one or more ITEM. Másik oldalról: Each ITEM must be PART OF one and only one ORDER.
Magyarul is viszonlag könnyű ilyen mondatot legyártani, de ott már ragokat kell használni, kiemelt szavak nem szerepelnek önállóan (mondatszerkezet szabványosítás nehezebb).
Magyar verzió: kapcsolat RE-TE (rendelés - tétel) egy névvel bír.

A modellkészítés lépesei:

1. Kezdő egyedtípusok kiválasztása (kezdő egyedtípusok kiválasztása – kezdő, mert még úgyis módosulni fog a halmaz, jönnek be még újak)
2. Közvetlen kapcsolatok meghatározása egy kezdődiagram megrajzolásával (csak a közvetlen kapcsolatokat tüntetjük fel)
3. Kapcsolattípusok fokának meghatározása („seprűk”)
4. n:m típusú kapcsolatok feloldása
5. Redundáns kapcsolatok eltávolítása (pl áthidalók…)
6. A kapcsolatok jellegének és egymáshoz való viszonyának elemzése (kötelező/esetleges kapcsolatok vizsgálata, kizáró kapcsolatok)
7. Modell ellenőrzése (megfogalmazott visszakeresési igény alapján történik. A követelmények kielégítését végigkövetjük a logikai adatmodellen)

A relációs adatelemzés

Célja: optimális adatszerkezet meghatározása (az adatszerkezet optimális, ha valósághű és ezt a legtakarékosabban oldja meg).
Adatmodell: nem azért adatmodell, mert az adatokat modellezi, hanem azért, mert a valós világ egy részét adatokkal írja le

Bővebben a *relációs adatelemzés célja:*
– megfogni a felhasználók részletes tudását az adatok jelentéséről és jelentőségéről
– ellenőrizni a logikai adatmodell érvényességét:
  - biztosítani, hogy a logikai adatmodell 3NF-ben legyen
  - biztosítani, hogy a logikai adatmodell megfeleljen a feldolgozási igényeknek
  - biztosítani, hogy a logikai adatmodell tartalmazza az igényelt részleteket
– biztosítani azt, hogy az adatok logikailag könnyen karbantarthatók és kiegészíthetők legyenek:

- biztosítani, hogy minden adatok közti függőséget biztosítsanak
- biztosítani, hogy a kétértelműséget feloldják
- megszüntetni a felesleges adatismétlődést

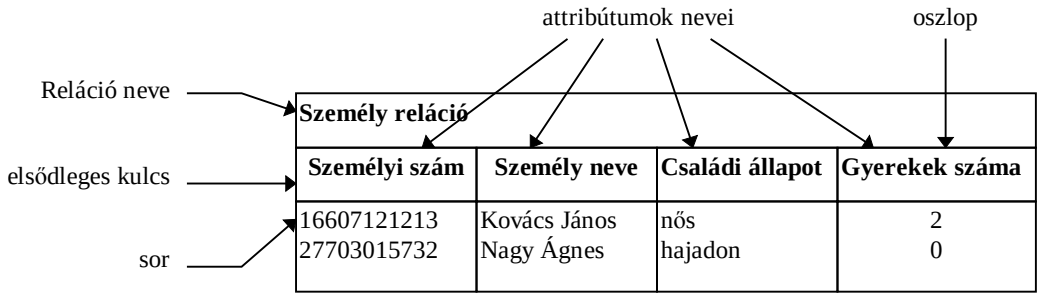
– optimális adatcsoportok kialakítása, amelyek alapot adnak az adatok különbözőalkalmazások közti felosztására.

Alapfogalmak:

A reláció: 2D-s táblázat. Minden egyes oszlop egy attribútumát jelenti az adott relációnak. Egy táblázatnak a következő tulajdonságokkal kell rendelkeznie ahhoz, hogy relációnak lehessen nevezni:

- nincs két egyforma sor
- a sorok sorrendjének nincs jelentősége
- az oszlopok sorrendjének nincs jelentősége
- minden oszlopnak egyedi neve van

Pl.:



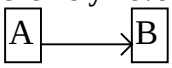
sor: egy adott objektumot jelöl
oszlop: attribútumok (tulajdonságtípusok)

Funkcionális függés: (két attribútum között definiáljuk)
B funkcionálisan függ A-tól, ha A egy konkrét értékéhez B egyetlen értéke rendelhető hozzá.

Pl.: A : személyi szám
B : személy neve

B **A**

személy neve funkcionálisan függ a **személyi számtól**



1 személyi számhoz egy név tartozhat, azaz a személyi szám meghatározza a személy nevét.

Kulcsok (egyedi azonosító)
Ezek azok az attribútum(ok), amelyektől más attribútumok funkcionálisan függnek. Tehát a kulcs attribútum egyértelműen meghatározza a reláció egy sorát.

Egy sorban lehet több olyan attribútum is, ami egyedi azonosító → ezek a kulcsjelöltek → ebből kell kiválasztani egyet → ez lesz az elsődleges kulcsa a relációnak.

- Ha egy jelölt kulcs egyetlen attribútumból áll, az az **egyszerű kulcs**
- Ha egy jelölt kulcs több attribútumból áll, az az **összetett kulcs**

- Ha a relációnkban előfordul egy másik reláció jelölt kulcsa, akkor ezt **idegen kulcsnak** (vagy **külső kulcsnak**) nevezzük
- Ha egy jelölt kulcsnak van egy vagy több idegen kulcs és egy nem idegen kulcs része, az a **hierarchikus kulcs.**

Pl.: **Megrendelés** ←reláció neve
Rendelésszám ←kulcshoz tartozó attribútum
Rendelés dátuma
Ügyintéző
Szállítási határidő
* Vevőkód ←olyan idegen kulcs, ami a kulcsunknak nem része

Rendelésszám, rendelés neve, ügyintéző, szállítási határidő, vevőkód : a reláció attribútumai (oszlopai, ha táblázatban jelöljük).

Normalizálás:olyan relációink lesznek, amelyekben minden nem-kulcs tulajdonságtípus a kulcstól, mégpedig - összetett kulcs esetén - annak egészétől függ funkcionálisan, és más függés nincs a relációban. Azokat a relációkat, ahol ez teljesül, **normalizált**-nak nevezzük.
Normalizált állapotban nincsenek anomáliák.

- **1NF:** ***(ismétlődő csoportok eltávolítása, mert az funkcionálisan független a kulcstól)*** A nem normalizált reláció úgy kerül 1NF-be, hogy eltávolítjuk belőle az ismétlődő csoportokat. Az olyan reláció, amelyben nincs ismétlődő csoport, eleve legalább első normál formában van. Az ismétlődő csoportot egyébként azért távolítottuk el, mert az funkcionálisan független a kulcstól, pontosan azért, mert ugyanazon kulcsérték mellett nem csak egy, hanem több különböző értéket felvevő adatokról van szó.
  Az ismétlődő csoportot nem egyszerűen levágjuk, mert akkor elveszne az információ, hanem mivel az ism. csoportot relációba ágyazott másik relációként is fel lehet fogni, ezért keresünk hozzá elsődleges kulcsot, és külön rakjuk. Így szedjük szét a könyvtári adatokat is. De így elveszne az író-könyv, és a kölcsönző-könyv kapcsolat, így a leválasztott relációknak összetett kulcsot kell adnunk.
  Általános szabály, hogy az ism. csoportot úgy választjuk le, hogy kijelölünk benne egy elsődleges kulcsot, amihez hozzátesszük az eredeti reláció elsődleges kulcsát, így összetett elsődleges kulcsot kapunk.
- **2NF:** ***(csak az összetett kulcsokat vizsgálni, a cél a részleges függés megszüntetése)*.** Ha egy relációban összetett elsődleges kulcsunk van, akkor könnyen előfordulhat, hogy valamelyik, bennük szereplő tulajdonságtípus a kulcs egyik részétől önállóan is függeni fog funkcionálisan. Nekünk pedig az a célunk, hogy az elsődleges kulcs egészétől függjenek csak a többiek. Az ilyen, részleges függést megvalósító tulajdonságtípusokból ismét új relációt képezünk. A példában szereplő táblák közül kettőben is találunk ilyet (alsó kettő). Így megint módosítunk.A második normál formához tehát az kell, hogy a reláció már eleve első normál formában legyen, és ne legyen benne részleges funkcionális függés.

Egyszerű kulcsok esetén az első és második normál forma megegyezik.

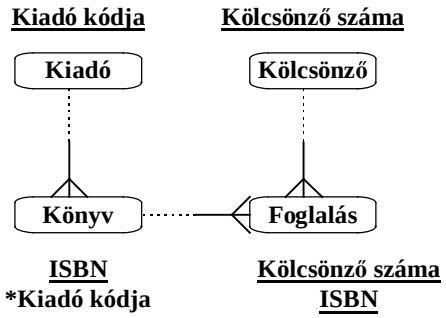
- **3NF:** ***(nem kulcs-jelölttől való függés eltávolítása)*.** Függetlenül a kulcs felépítésétől, bármelyik relációban előfordulhatnak a leíró tulajdonságtípusok közötti – belső – függések. Ezeket is ki akarjuk küszöbölni, hogy mindenki csak az elsődleges kulcstól függjön.
  Látható, hogy a második és harmadik tulajdonságtípus egyaránt az elsőtől függ, és a harmadik a másodiktól is. Ezt úgy hívjuk, hogy a harmadik a másodikon keresztül az elsőtől közvetetten, tranzitívan függ.Ezeket kell eltávolítanunk a harmadik NF-hez úgy, hogy szintén új relációkat generálunk. Példánkban a kiadó kódján keresztül függ a kiadó neve az elsődleges kulcstól. Látható, hogy a régi és új reláció között a kapcsolatot a Kiadó kódja

létesíti, amiből idegen kulcs lett. A harmadik normál forma feltétele tehát, hogy második normál formában legyen a reláció, és ne legyen benne belső függés.

Nem normalizált	1NF	2NF	3NF	
ISBN	**ISBN**	**ISBN**	**ISBN**	
Író neve	Író neve	Író neve	Író neve	KÖNYV
Könyv címe	Könyv címe	Könyv címe	Könyv címe	
Kiadás éve	Kiadás éve	Kiadás éve	Kiadás éve	
Kiadó kódja	Kiadó kódja	Kiadó kódja	*Kiadó kódja	
Kiadó neve	Kiadó neve	Kiadó neve		
Kölcsönző száma				KIADÓ
Foglalás dátuma	**ISBN**	**ISBN**	Kiadó kódja	
Kölcsönző neve	Kölcsönző száma	Kölcsönző száma	Kiadó neve	
Kölcsönző címe	Foglalás dátuma	Foglalás dátuma		
Kölcsönző telefonja	Kölcsönző neve			FOGLALÁS
	Kölcsönző címe	Kölcsönző száma	**ISBN**	
	Kölcsönző telefonja	Kölcsönző neve	Kölcsönző száma	
		Kölcsönző címe	Foglalás dátuma	
		Kölcsönző telefonja		
			Kölcsönző száma	KÖLCSÖNZŐ
			Kölcsönző neve	
			Kölcsönző címe	
			Kölcsönző telefonja	

3NF-ből előállíthatjuk a logikai adatmodellt

- relációk → egyedtípusok
- idegen kulcsok → fölérendelt egyedtípust határoznak meg
- összetett kulcsok összetevői idegen kulcsok
- hierarchikus kulcs legalább egy összetevője idegen kulcs



Relációkból diagram

Célunk, hogy ugyanolyan diagramot készítsünk, mint a logikai adatmodellezés egyed-kapcsolati modellje esetében. Ehhez a következő szabályokat vannak:

- A relációk lesznek az egyedtípusok
- Az idegen kulcsok fölérendelt egyedtípust határoznak meg
- Az összetett kulcsok összetevői idegen kulcsok
- A hierarchikus kulcs legalább egyik összetevője idegen kulcs

Példánkban: először az első szabályt alkalmazzuk:

KÖNYV
KÖLCSÖNZÖ
ÍRÓ
SZERZŐ
FOGLALÁS
KIADÓ

A 2. szabályt alkalmazva: a KÖNYV fölérendeltje a KIADÓ.
3. szabályból: a SZERZŐ-ben és a FOGLALÁS-ban két idegen kulcs van az összetett kulcsban, ezekre megint jöhet a 2. szabály…
A kapcsolatok jellegét is meg kell határozni. Annyi biztos, hogy az elsődleges kulcsokhoz tartozó tulajdonságtípusokhoz minden egyed-előfordulásban kötelező értékkel kell rendelkezniük, különben nem lehetnének kulcsok. A FOGLALÁS összetett kulcsára gondolva ez annyit jelent, hogy ennek az egyedtípusnak a szempontjából mindkét fölérendeltje kötelező…

Ha minden I/O szerkezetet elemeztünk, akkor tudunk olyan ábrát csinálni, mint amilyen a logikai adatmodell volt. Ezzel hasonlítjuk össze, és ha eltérés van, akkor azt ki kell vizsgálni.