BEVEZETÉS AZ ADATBÁZIS KEZELŐ RENDSZEREK ALKALMAZÁSÁBA

1.1. Ismétlő áttekintés

Ismeret – Adat – Információ

Az emberek (nem csak a gazdaság, a közigazgatás vagy egyéb tevékenységet folytató területek irányítói, résztvevői) arra törekszenek, hogy a környezetükből származó és tevékenységüket esetleg még meg is határozó **i s m e r e t** -eket valamilyen módon megőrizzék. Véges a memóriájuk és persze felejtenek is.

A megszerzett ismereteiket tehát el kell "raktározni" de ehhez valamilyen formában rögzíteni, esetleg még átalakítani is kell azokat. Az így átalakított, és rögzített ismeretet nevezzük **a d a t** -nak. Az adatok tárolásának módja az idők folyamán változott, jelenleg erre a célra a számítógép látszik legalkalmasabbnak. Az adat a valóság valamilyen objektumához kötődik, leírja annak tulajdonságát. Beszélhetünk még az **e l e m i a d a t** –ról, amely tovább már nem osztható, önmagában még értelmezhető ismeret. Azzal a példával lehet ezt leginkább érzékeltetni, hogy egy személyi nyilvántartás lakcím adata [Irányítószám, Helység, Utca név, Házszám, Emelet, Ajtó] elemi adatokból áll.

Az adatok eltárolásának azonban nem maga a tárolás a célja. Ez csak eszköz ahhoz, hogy a tárolt adatok gyors, gazdaságos feldolgozása által (szintetizálás, kiválogatás, elemzés) új ismerethez, azaz **i n f o r m á c i ó** -hoz jussunk. Az információ szintén erőforrás. Jellemzője, hogy nincs megmaradási törvénye, független az üzenet tartalmától és átvitelenként sérülhet.

Az elemi adatot az objektumról a **j e l** közvetíti, amelyhez az észlelést végző **jelentést** társít, s így jut információhoz. A jel fizikai megjelenési formája a kommunikáció fajtájától függ. Például a forgalomirányító lámpa piros fénye azt jelenti a közlekedők számára, hogy az adott irányba most tilos a továbbhaladás. A jel tehát egy „megállapodáson” alapuló információhordozó. A leírt (lásd betűk, számok), s témakörünkben maradva, a számítógépen tárolt jelsorozat (bináris értékek) a számítógépben tárolható. Feldolgozó mechanizmusa által értelmezhető s belőle információ állítható elő. Ennek megjelenési formája szintén megállapodáson alapul. A feldolgozás outputja lehet diagram (grafikus jel), számsorozat (numerikus jel), szöveg, kép, hang, stb.

Az adatok tárolása

Az adatokat természetesen azért tároljuk, hogy feldolgozhatóak legyenek. Egy adattároló rendszernek tehát minimum annak kell megfelelni, hogy adott tulajdonság alapján megtalálható legyen egy keresett adat. Ez nem jelent mást, mint, hogy előre meghatározott eljárásokkal (algoritmusokkal) hatékonyan feldolgozható legyen egy adathalmaz. Úgy is mondhatjuk, hogy meghatározott szerkezetben kell az adatokat tárolni.

A számítástechnikai feldolgozások az adatokat **á l l o m á n y** -okban (adatállományokban) tárolják. Ezeket nevezzük **f á j l** -nak is. Az adatállomány egy sora (logikai egysége) a **r e k o r d**, ennek alkotórészei az elemi adatok, a **m e z ő** -k.

Meg kell még említeni, különösen az adatbázis szemléletű adatkezelés kapcsán, az **a d a t c s o p o r t** fogalmát, amely valamilyen közös ismérv alapján együtt értelmezett elemi adatok (mezők) halmaza.

Az adatok feldolgozása

Az állománykezelő vagy szakszerűbben **f á j l k e z e l ő m ű v e l e t** -ek a fájlok, a rekordok illetve a mezők szintjén kerülnek meghatározásra.

Létrehozás - a fájlszerkezetnek a kialakítását jelenti, amely magában foglalja a tárolás logikai (fájl típus, rekord szerkezet) és fizikai paramétereinek (tárolt adatok típusa, mérete, tárolási hely) meghatározását

Karbantartás - a fájl tartalmának rekord illetve mező szintű módosítása.

Új Rekord felvitele
Rekord törlése - **Logikai törlés** (a rekord létező de inaktív)
- **Fizikai törlés** (a végleges megszüntetés)

Rekord módosítás, amely mező szintű művelet, a mező(k) tartalmának változtatása

Visszakeresés - az állomány rekordjaira illetve mezőire vonatkozó feltétel alapján történő adat keresés. A gyakorlatban ez a művelet egy mező szintű feltétel alapján, rekord szintű kereséssel indul és ha megtaláltuk a rekordot, kerülhet sor a szükséges feldolgozás (módosítás, törlés) indítására.

Újraszervezés - a fájl (logikai) szerkezetének átalakítása, esetleg tartalmi változások okán újra létrehozása. Ez a művelet általában technikai, karbantartó céllal történik és nem közvetlenül az adatfeldolgozás része.

A fájlszerkezetek, vagyis az alkalmazott adatállományok típusai:

Soros fájlszerkezet: a legegyszerűbb, de feldolgozás szempontjából legmunkaigényesebb szerkezet. A soros fájlban a rekordok minden rendezettség nélkül, a felvitelük sorrendjében kerülnek eltárolásra.
Az új rekord, a már létező rekord sorozat végére, utánírással vihető fel.
A törlés, módosítás csakis újraszervezéssel, vagyis a módosított tartalmú fájl felírásával és a módosítás előtti fájl megszüntetésével, törlésével hajtható végre.
A visszakeresés pedig a fájl tartalmának végigolvasásával lehetséges, amely mindaddig tart, amíg meg nem találjuk a keresett rekordot illetve az állomány végére nem jutunk.

Szekvenciális fájlszerkezet: a fájl rekordjai itt is sorosan érhetők el, de a rekordok sorrendje valamilyen felhasználói igény szerint már rendezettek. A kereséskor egy közvetlen elérésű adattárolón (merevlemez) gyorsabb elérési módszerek is alkalmazhatók, mint a rekordok végigolvasása, főleg több rekord egyidejű feldolgozása esetén.
A törlés, módosítás mellett az új rekord felvitele is a fájl újraszervezését igényli mert a rendezettség fenntartása érdekében (nem lehet kér rekord közé beszúrni).

Direkt (vagy közvetlen) hozzáférésű fájlszerkezet: jellemzője, hogy a rekordok és tárolási helyük fizikai vagy relatív (egymáshoz viszonyított) címei között rögzített, közvetlen kapcsolat van. A

cím vagy sorszám rekordhoz rendelése a feldolgozó program feladata, amely a felhasználó rekord elérési igényeit tükrözik. A gyors feldolgozási igények olyan feladatokra terjedhetnek ki, mint az 1. vagy utolsó vagy n. rekordra ugrás, minden 5. rekord feldolgozása, stb. A rekordok feldolgozhatók a címük sorrendjében és a háttértáron történő fizikai tárolásuk sorrendjében (mint egy soros állománynál).
A törlés speciális jelenséget okoz, az úgynevezett *álrekordot*, ami nem más, mint a fizikai sorban benne levő de tartalmát "vesztett" rekord. (lásd a logikai törlést) A hely és a helyhez tartozó cím megvan de nincs értékes adat benne. Az álrekordok törlése az állomány átszervezésével a fizikai törléssel történhet.
Az új rekord felvitele az ilyen álrekordok helyére történhet, vagy utánírással.

Indexelt szekvenciális fájlszerkezet: egy direkt elérésű alapfájlból és egy vagy több úgynevezett indextáblából áll. Az indextábla önmaga is egy állomány, amely két fontos adatot tartalmaz. Az egyik azon mező vagy mezők csoportja, amely(ek) szerint indexelni kívánjuk az alapállományt, a másik pedig az alapfájl rekordjainak cím adata. Az indexelés nem más, mint az adott mező(k) szerinti rendezettségi sorrend.
A visszakeres az indextábla alapján, a valódi műveletvégzés az indexfájlban tárolt cím segítségével gyorsan megtalált alapfájlon történik meg.
A törlés és új rekord felvitel természetesen az index táblák újraszervezését is jelentik.

A logikai és fizikai adattárolási modellek összefüggése

Az adatok logikai szerkezete (lásd az előző bekezdésben) és az adattárolón történő fizikai elhelyezkedése a hatékony hozzáférés biztosítása miatt összefügg. A fizikai tárolás problematikáját az Operációs rendszer oldja meg, de a fejlesztő szoftverek is gyakran rendelkeznek az adattárolás módjáról.

(Lásd a Programozás elmélet tankönyv B.2.2.3. fejezete.)

Irodalom: Információs Rendszerek Szervezési Módszertana ComputerBooks, Budapest, 1993.
Adatbázis-kezelés - Bakó Sándor, Pedellus Novitas Kft kiadó, Debrecen, 1999.

1.2. AZ ADATBÁZIS KEZELÉS ALAPFOGALMAI

1.21. Az adatbázis fogalma

Az ADATBÁZIS, logikailag összefüggő, meghatározott szerkezetben tárolt adatok halmaza.

Egy adathalmaz önmagában **még nem adatbázis**, mert hiába tartalmaz sok adatot, azok között hiába ismerhető fel esetleg logikai összefüggés, ha olyan kötetlen szerkezetű, amelyben semmilyen szabályszerűség nem ismerhető fel és így nem elemezhető egy előre meghatározott szempont szerint. Ilyen lehet például egy leltárhiányt rögzítő jegyzőkönyv (hol, mikor, ki vette fel, mik a megállapításai, vagy egy olyan lista, amelyben emberek minden lehetséges adatát szeretnénk nyilvántartani.

Az egyik probléma, hogy annyiféle információ kapcsolható egy személyhez, hogy egyetlen adattárban elhelyezve áttekinthetetlen és így együtt felesleges is lesz (név, lakcímek, különféle dátumok, testfelépítés adatai, egészségügyi adatok, foglakozási adatok, adóhatósági adatok, párkapcsolatok, stb.)

A másik probléma pedig, hogy egy "életszerű" feldolgozás, vagyis egy valós információ igény kielégítése mindig valamilyen logikailag összefüggő adat csoportra vonatkozik (egészségbiztosítási, lakcím, iskolai végzettség, stb.)

Adatbázisnak tekinthető viszont minden, a feldolgozási (információ nyerési) igény szerint összeállított, úgynevezett **strukturált adattömeg**, függetlenül annak tárolási módjától. Mint például az iratgyűjtőbe, dátum szerint lefűzött levelek, a telefonkönyv, egy menetrend vagy az osztály tanulóinaknak adatai a naplóban.

A számítógépen tárolt adatbázis viszont egy olyan rendezett adathalmaz, amely amellett, hogy a feldolgozásnak megfelelő logikai struktúrába van szervezve, még a számítógépi tárolás kritériumainak is megfelel. Úgymint rekordszerkezeti és gépi feldolgozhatóság kritériumai.

Az adatfeldolgozás és így nyilvánvalóan az adatbázisként történő feldolgozás **környezetünk tetszőleges objektumai**ra vonatkozhat, amelyeket **EGYED** –nek nevezünk. Az egyed "megjelenési formája" számítógépes környezetben egy "valamilyen" szerkezetű fájl illetve ezen fájl rekordjai.

Az egyedek **jellemző tulajdonságok**kal, **ATTRIBÚTUM** -okkal rendelkeznek, amelyeket tárolunk, feldolgozunk, átalakítunk valamint általuk kapcsolatokat teremtünk az egyedek között.

pl. A videó kölcsönző forgalmát feldolgozó rendszer egyedei a videofilmek (adatai) és a kölcsönző személyek (adatai). Az ezekhez tartozó tulajdonságok pedig a <Film címe, Műfaja, Hossza, Kölcsönzési díja, stb.> illetve a < A kölcsön vevő személy neve, Lakcíme, Mikor kölcsönözte ki a filmet, stb.>

1.22. Az Adatbázis Szemlélet

Az előzőekből látható, hogy az adatbázis nem egyszerűen egy adathalmaz, hanem egy, a feldolgozási célnak alárendelt szerkezetű, kötött adatstruktúra. Ezt a sokszor igen nagy és bonyolult adathalmazt ugyanakkor hatékonyan, gyorsan és hibamentesen kel feldolgozni. Az ilyen kritériumoknak azonban csak egy olyan programrendszer képes megfelelni, amely különböző feldolgozási körülmények között, sok és sokféle adat feldolgozására képes.

A hagyományos program-rendszerek kivitelezése során jelentős munkát kell befektetni az adatok szerkezetének, tárolásának és feldolgozásának definiálásába. Az adatbázis kezelő rendszerek azonban függetlenítik az adatot a programtól, ami azt jelenti, hogy nem a program utasításai rögzítik az adatszerkezetet, hanem maga az adatállomány rendelkezik olyan tulajdonságokkal, amelyek a saját szerkezetét meghatározzák.

Ezt nevezi az adatbázis elmélet **a d a t f ü g g e t l e n s é g** -nek amely egyrészt **Fizikai** adatfüggetlenség, tehát hardveres adattárolási, másrészt **Logikai** adatfüggetlenség azaz adatszerkezeti változástól függetleníti magát az adatbázis kezelő szoftvert. Ez a "függetlenítés" azt is jelenti, hogy lehetőség van egy adott programozási környezetben (adatbázis kezelővel) létrehozni az adatbázist, majd az üzemeltetés során egy másik adatbázis kezelővel fogjuk azt feldolgozni.

Az Adatbázis Szemléletű adatfeldolgozás jellemzői:

1. Érvényesül az adatfüggetlenség elve
2. Nem tárolunk többszörösen egy adatot (nincs redundancia, lásd később)
3. A tárolt adatok ellentmondásmentesek (nincs inkonzisztencia, lásd később)
4. Az adatbázishoz több alkalmazás és felhasználó is hozzáférhet (osztott adatbázis)
5. A többféle felhasználó igényt kielégítő adathalmaz egy központi helyen, közösen tárolható (az AB integrált)
6. Biztosított az adatok védelme

1.23. AdatBázis Kezelő Rendszerek (ABKR)

ADATBÁZIS FELDOLGOZÁS -ról beszélünk akkor, amikor egy nagytömegű, kötött szerkezetű adathalmaz (az adatbázis) feldolgozását végezzünk számítógéppel támogatott **AdatBázis Kezelő Rendszer (ABKR)** keretében.

Az **ABKR**, a számítógépen tárolt Adatbázisok létrehozását, karbantartását, feldolgozását biztosító, valamint az adatok sérülését megakadályozó szoftverek összefoglaló neve.
angolul: **DBMS** - **D**ata**B**ase **M**anagement **S**ystem

AB Kezelő szoftverek: dBASE, Oracle, MS SQL Server, MS ACCESS, Clipper, FoxPro

Az ABKR szolgáltatásai:
- az adatbázis szerkezetének létrehozása,
- strukturált (meghatározott szerkezetű) adattárolás,
- szabályozott adat karbantartás (bővítés, módosítás, törlés)
- adatrendezés (adott szempont szerinti sorba állítás)
- adatszűrés (adott szempont szerinti kiválogatás)
- szerkesztett lekérdezés (interaktív és nyomtatott)
- egyszerű listázás
- adat export és import (kapcsolat más adatbázisokkal)
- adatvédelmi eljárások

Az ABKR -ek függetlenséget biztosítanak

- Az adatkezelés fizikai, azaz HW lehetőségeitől
- Az adatelérés módjától
- Az adatszerkezettől

A felsorolt 3 "függetlenség" röviden a következőket jelenti:

Egy adatbázis és a hozzá kapcsolódó feldolgozó eljárások attól függetlenül legyenek kialakíthatók, hogy a feldolgozást végző hardver eszközök (processzor, háttértár, nyomtató, stb.) éppen milyen típusú, illetve a hardvert működtető alapszoftverek milyenek.

Az adatok elérésének és tárolásának módja (lásd fájlszerkezetek) különböző lehet. Az adatbázis kezelő szoftvereknek azonban még a rendszert fejlesztő programozó előtt is "el kell fednie" azokat az eljárásokat, amelyek által hozzájut a kívánt tartalmú és szerkezetű adatokhoz. A fejlesztőnek csak logikai adatkapcsolatokat kell meghatározni illetve a feldolgozás során csak a szükséges kérdéseket kell megfogalmazni a kívánt információ érdekében, de az eredmény előállításának módját nem kell definiálnia.

Az adatbázis szerkezete nem állandó. Új felhasználói igények merülnek fel az üzemeltetés során. Ezek többnyire további input adatok bevitelét és újabb output szolgáltatást (új listák, képernyők, stb.) jelentenek. Mindez az adatszerkezet módosulását vonja maga után. Ugyanakkor a "régi" feldolgozást változatlan formában igénylik. Egy új igény beépítése hagyományos rendszerben komoly program és adatállomány rekonstrukciót igényel. Az adatbázis kezelő rendszereknek azonban fontos kritériuma, hogy a változtatás ne érintse a korábbi feldolgozó eljárásokat bár az adatok szerkezete természetesen megváltozott.

Az ABKR -kel szembeni elvárások:

1. Az adatok definiálása különüljön el az adatkezeléstől, (ADATKATALÓGUS –ban történik)
2. Az adatkatalógus maga is az AB része, az adatokkal azonos módon történjen a feldolgozása
3. Az adatdefiníciónak, az adatkezelésnek és az adatbiztonságnak legyen programozási nyelve.
4. Ezen nyelvi eszközök adjanak módot más alkalmazások, programnyelvek számára a hozzáféréshez. (interfész elemek)

Az adatbázisokat kezelő programozási nyelvek tagolódása:

A korábbi megállapításokból következően, (lásd adatfüggetlenség, az adatvédelem fontossága, az elvárások 3. pontja) belátható, hogy az adatbázisok feldolgozása sok funkcióból, eljárásból tevődik össze. Ennek megfelelően az egyes funkciók megvalósítása érdekében különböző fejlesztő intézetek, különböző programrendszereket hoztak létre. Ezek a programrendszerek vagy lefedik az összes funkciót, vagy egy-egy részterület feladatait oldják meg. A fontos csak az, hogy ezek kompatibilisek legyenek egymással. Ezért nemzetközi konferenciákon közösen kialakított, szabványosítási megállapodásokkal biztosítják az "átjárhatóságot" a programrendszerek között. A szabványosítási törekvések eredményeként születtek az egyes részterületeket lefedő önálló programnyelvek.

1. Az Adatleíró nyelv (**D**ata **D**efinition **L**anguage - **DDL**)

A számítógéppel feldolgozott adatok tulajdonképpen a valós világ objektumainak valamilyen leképezése, modellezése. Ez az **adatmodell** szolgál a felhasználó által igényelt információk alapjául. Első lépés tehát a rendszer létrehozása során a feldolgozásra alkalmas adatmodell kialakítása. Nevezzük még ezt a logikai modellt **sémá** -nak is. Egy integrált, osztottan alkalmazott adatbázis esetén az egyes felhasználók által „látott” adatszerkezet (modell) neve pedig **alséma**, amely a teljes adatbázisnak az adott felhasználói igény szerinti részhalmazát írja le.

Az adatmodell (séma) létrehozása során tartalmi és formai előírásokat rendelünk a tárolandó adatokhoz. Ennek megfogalmazására szolgál az **Adatleíró nyelv**, ami valójában nem más, mint program utasítások csoportja, amelyekkel definiáljuk az adatok nevét, adattípusát, méretét, formátumát, hozzáférhetőségét. Azt a "listát", amely a használt adatok jellemzőit tartalmazza, **Adatszótár** –nak nevezzük.

Megjegyzés: ilyen utasítások például a CREAT <objektum>, ALTER <objektum> (lásd SQL jegyzet)

2. Az Adatkezelő nyelv (**D**ata **M**anipulation **L**anguage - **DML**)

Az adatbázisok feldolgozása az úgynevezett **adatbázis-kezelő műveletek** sorozataként valósul meg. Ez azt jelenti, hogy az adatbázis elemeket (állományokat, rekordokat, kapcsolat leíró elemeket, stb.) és az adatokat mozgatni, létrehozni, felvinni, keresni, módosítani, törölni, egyszóval manipulálni (kezelni) kell.

Az adatbázis-kezelő rendszerek erre a célra rendelkeznek saját parancskészlettel (DML), amelyet önállóan alkalmaznak a feldolgozás során, de lehetséges az is, hogy a DML beépül egy magas szintű programozási nyelvbe és annak utasításai között kerülnek meghívásra és végrehajtásra.

Megjegyzés: ilyen utasítások például az INSERT INTO …, UPDATE …, (lásd SQL jegyzet)

3. A Lekérdező nyelv (**Q**uery **L**anguage - **QL**)

Az adatbázisban tárolt adatok legfőbb értéke, hogy belőlük, meghatározott igények szerint adatok, végső soron információk nyerhetők. A „meghatározott” adatnyerés leegyszerűsítve annyit jelent, hogy a teljes adathalmazból csak egy részhalmazra vagyunk kíváncsiak, és/vagy az adatokat valamilyen sorrendben szeretnénk látni, és/vagy olyan adatokat akarunk együtt látni, amelyek az adatbázisban nem tárolódnak együtt.

Az adatbázis-kezelő rendszerek erre a célra szintén rendelkeznek saját parancskészlettel (QL), amely olyan utasításokat foglal magában, amelyek ezeket az igényeket kielégítik. A lekérdező nyelv szintén beépülhet egy magas szintű programozási nyelvbe.

A lekérdezés kiemelt szerepét jellemzi, hogy ez a terület az, amelyet a szoftver gyártók megegyezése alapján standardizálni (szabványosítani) lehetett. Ez a szabványos lekérdező nyelv az **SQL** - **S**tructured **Q**uery **L**anguage, azaz magyarul a Strukturált lekérdező nyelv.

Megjegyzés: ilyen utasítások például az SELECT …, WHERE …, GROUP BY … (lásd SQL jegyzet)

4. Az Adatvezérlő nyelv (**D**ata **C**ontrol **L**anguage - **DCL**)

Az adatbázis feldolgozása során sok és bonyolult művelet, idegen szóval **tranzakció** kerül végrehajtásra. Ahhoz, hogy az adatbázis egy ilyen tranzakció után is „jó” maradjon, ne sérülhessenek benne adatok vagy adat kapcsolatok, folyamatosan ellenőrizni (kontrollálni) kell a műveletvégzéseket. Biztosnak kell lenni abban, hogy valóban végrehajtódott-e a kívánt művelet, minden szükséges művelet megtörtént-e és abban a sorrendben ahogy kell, nem szakadt-e meg egy művelet lánc, stb. Ugyanakkor azzal a minimális biztonsággal is rendelkeznünk kell, hogy ha véletlenül nem jól kerültek végrehajtásra a műveletek vagy a kezelő hibájából „véletlenül” egy nem kívánt művelet (például törlés) került végrehajtásra, az eredeti állapot visszaállítható legyen. De hasonló problémákat okozhatnak a géphibák vagy a program téves működése.

Még egy példa a tranzakció értelmezésére, amely egymással összefüggő műveletsort takar. A tranzakció elkezdődik (BEGIN) amikor valaki a bank betét ablakánál megjelenik és közli, hogy pénzt kíván elhelyezni a számláján. Következik a QL utasítás, amellyel lekérdezik, hogy a személy létező ügyfél-e és azonosítható-e a bemutatott okmányok alapján. Ezután módosítják a számla adatát a betét összegével, amely egy DML utasítás. Mi van akkor, ha az ügyfél a pénztárnál jön rá, hogy otthon hagyta a pénzét? Minden eddigi műveletet törölni kell, pontosabban a műveletek hatását az adatbázisból. Szükség van tehát egy olyan utasításra, amellyel nyugtázni lehet a műveletek elvégzését a BEGIN –től (COMMIT) vagy érvénytelenít mindent (ROLLBACK). Ez után következhet csak az újabb „betét” tranzakció.

Megjegyzés: a BEGIN, COMMIT, ROLLBACK az adatvezérlő nyelv utasításai.

5. Az Adatvédelmi utasítások

Egy adatbázis, a benne tárolt adatok egyrészt anyagi értéket képviselnek, pontosabban a megsemmisülése vagy illetéktelen kezekbe kerülése anyagi kárt okoz. Gondoljunk továbbá egy kórházi adatbázisra ahol emberek „életét jelentő” adatok tárolódnak. Bár nem beszélünk adatvédelmi nyelvről, de külön beszélni kell az adatvédelmi utasításokról. Itt csak a figyelem felkeltése a cél, de az SQL tárgyalása során részletesen foglakozunk vele.

Két alapvető célt valósítanak meg az adatvédelmi utasítások: **hozzáférési jogokat** definiálnak az adatbázishoz és annak elemeihez illetve a **műveletvégzést** szabályozzák. Például egy adatbázis bizonyos adatcsoportjához férhet csak hozzá az adott kezelő illetve amihez hozzáfér azt is csak olvashatja és módosíthatja de már nem törölheti.

1.24. Az Adatfeldolgozó Információs Rendszerek tervezésének és létrehozásának szakaszai

- A felhasználói igények és a rendelkezésre álló erőforrások (HW, SW, MW) felmérése
- Az igények és a lehetőségek egyeztetése
- A feldolgozáshoz szükséges adatok körének meghatározása
- **Az adatok tulajdonságának meghatározása**
- **Az adatok kapcsolatának meghatározása**
- **Az adatbázis LOGIKAI tervezése**
- **Az adatbázis FIZIKAI tervezése**
- Az adatfeldolgozó folyamat tervezése
- Az üzemeltetés feltételeinek meghatározása
- A kész rendszer próbája, tesztelése, javítása

A fenti felsorolás kiemelt soraiban szereplő tevékenységek az adatbázis szemléletű adatfeldolgozás tervezése és kivitelezése során különös fontossággal bírnak. Jegyzetünk ezek részletes tárgyalásával folytatódik.

A Rendszerszervezés súlypontja itt az adatok tartalmi és kapcsolati rendszerének meghatározása.
(Lásd részletesen az Információtechnológia jegyzet 1.42. fejezetében.)

Logikai adatszerkezet majd fizikai modellek készülnek, amelyek beépülnek az adott ABKR egységesített feldolgozó rendszerébe.

A felhasználói folyamat modell alapján pedig egy keretprogram látja el a felhasználói igények kielégítését. (Az AB kezelő nem algoritmikus nyelv !)

Az ABKR –ek nem képesek algoritmikus feladatok végrehajtására, tehát elágazások, ciklusok, egyéb összetett utasítások végrehajtására.

Az adatfeldolgozás ilyen jellegű feladatait egy **k e r e t p r o g r a m** végzi el.

A Keretprogram szerepe

Az adatfeldolgozási feladatot, a szükséges algoritmus alapján az úgynevezett **Keretprogram** oldja meg, amely magába integrálja az ABKR utasításait. Ekkor mondjuk, hogy az adatbázis kezelő utasítások egy magas szintű programnyelvbe beágyazottan kerülnek felhasználásra.

A keretprogram oldja meg tehát a lényegi adatfeldolgozást, az adatbevitel, a tárolás, adatmentés, visszatöltés feladatait output szolgáltatás feladatait.

A bonyolult adatállomány kezelési (létrehozás, módosítás, törlés) és lekérdezési (szűrés, rendezés, listázás) feladatokat viszont az ABKR látja el leghatékonyabban.

1.25. AZ ADATTÁROLÁS MÓDJA ADATBÁZISOKBAN, AZ AB SZERKEZETE

- Az **ADATBÁZIS**, ADATÁLLOMÁNYOK (adatfájlok) halmaza.

Pl. Egy Videó kölcsönző forgalmát feldolgozó AB a ***Videofilmek*** és a ***Kölcsönző személyek*** adatait tartalmazó adatállományokból áll.

- Az **ADATÁLLOMÁNY**, logikailag összetartozó adatok meghatározott (előre megtervezett) szerkezetű tárolási formája. Más néven ADAT FILE.

Pl. A ***Videofilmek*** adatállománya a Videotékában forgalmazott filmek *Cím*ét, a megjelenés *Dátum*át, a film *Műfaj*át, *Kölcsönzési díj*át, stb. tartalmazza.

A ***Kölcsönző személyek*** adatállomány a Videotékából kölcsönző ügyfelek *Személyi igazolvány szám*át, *Nev*ét, *Lakcím*ét, *Születési dátum*át, stb. tartalmazza.

- Az adatállomány **REKORD** -okból áll, amelyek egy-egy konkrét adatsort tartalmaznak.

Az előbb felsorolt adatok, (*Műfaj, Név, Lakcím*, stb.) nem konkrét tartalmak, csak az adatok nevei, amelyekkel hivatkozunk rájuk. Egy adott film vagy ügyfél adatai az adatállomány rekordjaiban kerülnek eltárolásra. Minden rekordban egy-egy konkrét film vagy személy összetartozó adatai szerepelnek. Fontos jellegzetesség, hogy az adatok sorrendje minden rekordban azonos. Ezt értjük a "meghatározott szerkezetű" azaz strukturált adattárolás alatt.

Pl. Egy film rekordja: Star Wars / 2000 / Scifi / 100 Ft
Egy ügyfél rekordja: ES 672 451 / Mészáros István / Puskás T. u. 13. / 1981.01.22.

- A rekord eleme a **MEZŐ** (field), amely egy konkrét adatot jelent az adatsorozatból.

Pl. A mező neve: ***Cím***, tartalma: Star Wars 1. vagy a mező neve: ***Lakcím***, tartalma: Puskás T. u. 13.

Az adatbázis szerkezete tehát az alábbi hierarchikus modellel írható le:

ADATBÁZIS { ADATÁLLOMÁNYOK { REKORDOK { MEZŐK } } }

Megjegyzés: a hagyományos (nem adatbázis szemléletű adatfeldolgozás adattárolási modellje is hasonló, csak elmarad az első, ADATBÁZIS elem. A hagyományos adatfeldolgozás is adatállományokat kezel de ezeket a feldolgozó program kapcsolja össze.

1.3. AZ ADATBÁZIS LÉNYEGE: AZ ADAT KAPCSOLATOK

Az adatbázist, az egymással jól definiált kapcsolatban álló adatállományok alkotják. Ez az előre megtervezett KAPCSOLAT - rendszer az, ami az adatbázis lényegét jelenti.

Pl. A videofilmek és a személyek adatait tartalmazó adatállományok csak akkor alkotnak adatbázist, ha közöttük kapcsolat mutatható ki. A fenti példa még nem "igazi" adatbázis, csak 2 adatállomány. Akkor lesz adatbázis, ha a ***Kölcsönző személyek*** adatállományban szerepel annak a filmnek a *Cím*e, amelyet kikölcsönzött az adott személy. Ezért a film *Cím*ét fel kell venni a személy adatai után a rekordban (esetenként többet is). Tehát mindkét adatállományban szerepelni fog a *Cím* mező, amely a két állomány kapcsolatát biztosítja.

A kapcsolat lehet: **1 : 1 fokú** kapcsolat
1 : N fokú kapcsolat
N : M fokú kapcsolat

A **KAPCSOLAT FOKA** azt jelenti, hogy az egyik adatállomány rekordja(i) hány rekorddal áll(nak) kapcsolatban a másik adatállományban.

Pl. Egy Személy általános adatait és személyiségi jogainak védelme érdekében, az egészségügyi adatait külön-külön adatállományban tároljuk. Biztosítani kell azonban, hogy ezeket az adatokat, az erre jogosult (például a háziorvos) szükség esetén együtt is kezelni tudja. Az (1) konkrét személy általános és ugyanazon (1) személy egészségügyi adatait leíró adatrekord között tehát nyilvánvaló összefüggés van. Ebben az esetben a két állomány között **1 : 1 -es kapcsolat** áll fenn. A kapcsolatot biztosító adat (mező) ebben az esetben lehet például a Személy neve vagy a TAJ száma. A lényeg az, hogy ez az adat mindkét állományban szerepeljen.

Bárkinek a tulajdonában egyszerre több ingatlan is lehet. Amikor a Személy adatait és a tulajdonában lévő ingatlanok adatait együtt is meg kívánjuk jeleníteni, akkor szükség van arra, hogy kapcsolatot biztosítsunk (1) konkrét személy és több (N) ingatlant leíró adatrekord között. Ebben az esetben beszélünk **1 : N –es kapcsolat** -ról. A kapcsolatot biztosíthatjuk úgy, hogy a Személy adatai között felsoroljuk a tulajdonában levő ingatlanok valamilyen azonosító adatát (ilyen lehet például az ingatlan Helyrajzi száma), vagy inkább (és ez a gyakoribb) az ingatlanok adatait kiegészítjük tulajdonosának azonosítójával, mondjuk a nevével, vagy a személyi számával.

Sokan úgy oldják meg a nyaralásukat, hogy bérleti jogot vásárolnak egy direkt ilyen célból épült üdülőtelepen. Ilyenkor egy apartmant tulajdonképpen többen, többen (N) közösen vásárolnak meg és elosztják egymás között az üdülési időpontokat. Az üdülőtelep fenntartójának ugyanakkor több
(M) ilyen apartman is a hatáskörébe tartozik. A fenntartónak tehát egyrészt ismernie kell a tulajdonosok adatait (értesítenie kell őket, számláznia kell) másrészt viszont az egyes apartmanok adataival is tisztában kell lennie (vízfogyasztás, a berendezések állapota). A két adatcsoportot nyilvánvalóan kapcsolatba kell hoznia a fenntartónak, mert például a felmerült közüzemi díjakat, a felújítási költséget a tulajdonosoktól kell beszednie. Az így kialakult kapcsolatot nevezzük **N : M –es kapcsolat** -nak. Az adatállományok kapcsolatának biztosítása ebben az esetben nehezebb, mint az előző két példában, de a későbbi tanulmányok majd erre is választ adnak.

Az adatállományok kapcsolatát tehát az adatállományok rekordjainak kitüntetett jelentőségű adata (mezője) biztosítja. Ezt a különleges adatot **K U L C S** –nak (**kulcsmező** -nek) nevezzük.

1.4. AZ EGYEDI AZONOSÍTÓ

Az EGYEDI AZONOSÍTÓ egy kiemelt jelentőségű adat, amely a rekordok egyediségét, a rekordra (annak adataira) való egyértelmű hivatkozás lehetőségét biztosítja. Egyedi azonosító (vagy rövidebben AZONOSÍTÓ) bármilyen adat lehet, amely a fenti feltételt teljesíti. A gyakorlat azonban többnyire az, hogy az AB tervezése, ezen belül az adatállomány tartalmának meghatározása során a Rendszerszervező kialakít egy megfelelő EGYEDI AZONOSÍTÓT, amely célszerűen biztosítja a rekord egyedi azonosíthatóságát. Az EGYEDI AZONOSÍTÓ bárhol elhelyezkedhet, a rekord adatai között, de célszerű azt az 1. mezőben elhelyezni.

Megjegyzés: későbbi tanulmányok során látható lesz, hogy egyedi azonosító nem csak egy mező lehet, hanem több mező együtt is képezhet azonosítót.

Pl. Nem véletlenül jöttek létre a személyeket (Személyi szám, Személyi igazolvány szám, Adóigazgatásai szám, TAJ szám), gépkocsikat (Rendszám), könyveket (ISBN szám) azonosító számok vagy betűk és számok kombinációi, egyszóval az Azonosító kódok. Mindegyik kód azt teszi lehetővé, hogy rá hivatkozva (azt feltüntetve) egyértelműen azonosítható legyen egy személy, egy autó vagy a könyv. Végső soron tehát, ha ezen objektumok adatait egy adatállományban tárolni akarjuk, akkor a rekordokban szerepelnie kell az objektum (egy személy, autó vagy könyv) adatsorát a többitől egyértelműen elkülönítő AZONOSÍTÓ –nak is.

Az egyedi azonosító lehet az azonosítandó objektum egy létező adata, amelyet

TERMÉSZETES AZONOSÍTÓ -nak nevezünk

és lehet az adatbázist létrehozó szakember által kialakított **AZONOSÍTÓ KÓD**.

Pl. Az előző példák azonosítói közül TERMÉSZETES AZONOSÍTÓ –nak tekinthető a Rendszám, mert az végső soron a gépkocsi egyik adata (ezzel együtt vásároljuk). Teljesen egyértelmű, hogy Magyarországon nem lehet kiadni két azonos rendszámot, s így az biztosan nem ismétlődhet, tehát az autót egyértelműen azonosítja. Ugyanakkor kézenfekvő, hogy az autó márkaneve, mint annak egyik jellemző adata már nem lehet azonosító, mert több nyilvántartott autónak is azonos lehet a márkája.
A személyeket és a könyveket viszont már ténylegesen AZONOSÍTÓ KÓD –okkal lehet csak megkülönböztetni. Sem a személynév sem a könyv címe - mint az objektum természetes adata - nem lehet egyedi azonosító, mert létezik több azonos nevű ember és egy könyvet többször is kiadtak sőt előfordul azonos című könyv is a világirodalomban.

A személyek azonosíthatóságával tehát gond van. Attól függően, hogy milyen célból illetve milyen nagyságú embercsoport tagjainak egyedi azonosítására van szükség, eltérő lehet a konkrét személyt azonosító kód.

Első megközelítésben egy személyt a neve azonosít. A nevet is kétfelé választhatjuk (Vezetéknév és Utónév), amelyek bizonyos körben és feltételek mellett külön is megfelelhetnek az egyediség követelményének. Az Utónév csak a testvéreket különbözteti meg egyértelműen a családban. A Vezetéknév, mint azonosító is erősen korlátozott. Csak olyan kis csoportokban lehet megfelelő, ahol bizonyosan nincs két azonos Vezetéknevű. (Azonos Vezetéknév még egy iskolai osztályban is gyakori előfordul, az Utónév egyezőség pedig még gyakoribb, sőt a teljes Név egyezőség sem ritka). Még a családban is lehetséges a teljes név azonosság, amikor apa és fia azonos Utónevet kapott.

A személyek azonosítására két módszer látszik lehetségesnek. Keresünk olyan adato(ka)t a személy adatai között, amely már megkülönböztethetővé teszi őket. Ennek egyik lehetséges logikai sorrendje lehet a következő: Vezetéknév < Utónév < Lakcím < Idősb/Ifjú előtét a névben >>> Ez a megoldás már majdnem jó, de mi van akkor, ha a Nagypapát is ugyanúgy hívják mint az Apát és a Fiút és ráadásul egy lakásban élnek ?

A számítógépes feldolgozás az előzőnél sokkal egyszerűbb azonosítást kíván meg és egy megfelelő Azonosító kódot alakítunk ki. A személyek esetében ennek még egy igen fontos vonatkozása is van, nevezetesen az ADATVÉDELEM. A személy azonosítása tehát kódolt. Csak megfelelő feldolgozási eljárással lehet megtudni, hogy az adott TAJ szám (és a hozzá tartozó egészségügyi adatsor) kihez tartozik. Még fontosabb, hogy a különböző tartalmú adatcsoportok, különböző kódokon elkülönülnek egymástól. A Személyi szám a Népesség nyilvántartás (népszámlálás, népesség statisztikák), a TAJ szám az egészségügyi adatok feldolgozása, az Adóigazgatási szám az adóhatóság tevékenysége, a Személyi igazolvány szám pedig például a lakóhely nyilvántartása során kerül alkalmazásra. Törvény írja elő, hogy ezek a külön-külön nyilvántartott adatok csakis büntetőeljárás során lehessenek összekapcsolhatók.

Fontos még az AZONOSÍTÓ KÓD –ok kialakításának módszere. Tulajdonképpen másra nem is kellene figyelni, mint, hogy biztosított legyen az egyediség (nem ismétlődhet) elvének érvényesülése. A számítástechnikai feldolgozhatóság azonban különleges igényekkel lép fel.

Ilyenek az **azonos hossz, azonos szerkezet, könnyű megjegyezhetőség, a tévesztés lehetőségének kiküszöbölése**. A felsorolás sorrendje egyben a fontosság növekedését és a feltétel érvényesítésének bonyolultságát is jelenti.

A **legegyszerűbb** egyedi azonosítás **a rekordok sorszámozása**, amely sok adatbázis kezelőben automatikusan beépített. Ebben az esetben biztosan nem ismétlődik az azonosító.

A Rendszerszervező azonban többnyire nem elégszik meg ezzel és saját egyedi azonosító kódot tervez az azonos hossz, szerkezet és megjegyezhetőség kritériumának kielégítésére.

Ennek eredménye lehet az úgynevezett **BESZÉLŐ KÓD**, amely szerkezetével és tartalmával utal arra az adattartalomra, amelyet azonosít és egyben "ránézésre" tájékoztat arról is, hogy mit azonosít.

Pl. A beszélő kód jó példája Személyi szám, amelynek 1. száma az illető személy nemét 2.- 7. száma pedig a születési dátumát tartalmazza.

Egy Híradástechnikai eszközöket értékesítő cég pedig TV9999999, RA9999999, VM9999999 szerkezetű Saját nyilvántartási kóddal azonosíthatja a Televízió, Rádió illetve Videomagnó készletét.

Az adatfeldolgozás egyik fontos kérdése a tévesztések kiküszöbölése. Ha megadunk egy Személyi számot, akkor biztosan annak a személynek az adatait akarjuk visszakapni a feldolgozástól, akiét szeretnénk és nem azét, akinek a Személyi száma a tévesztés miatt csak "majdnem ugyanaz". Ennek a feladatnak a megoldása elég nehéz, de nem lehetetlen.

A feladatot az úgynevezett **E L L E N Ő R Z Ő S Z Á M** alkalmazásával oldják meg.

Pl. A Személyi szám beírása során történő tévesztés elkerülése érdekében a Személyi szám 11. pozícióján szereplő ELLENŐRZŐ SZÁM –ot alkalmazzák.

A Személyi szám szerkezete: **9 ÉÉHHNN 999 9**

- **9** – A személy neme: 1: férfi, 2: nő
- **ÉÉHHNN** – Születési dátum
- **999** – Az egy napon születettek sorszáma
- **9** – Az ELLENŐRZŐ SZÁM

Az ELLENŐRZŐ SZÁM funkciója az, hogy minden egyes beírás esetén ellenőrizze, hogy a Személyi szám minden értéke jó-e. Ezt úgy lehet elérni, hogy a beírás után egy azonos algoritmus szerint újból kiszámítják az ellenőrző számot és ha az egyezik a 11. beírt számmal, akkor az előző 10 számot sem írtuk be rosszul. Ezt az ellenőrző számot akkor kapja a Személyi szám, amikor először létrehoznak egyet a születés bejelentésekor az Állami Népesség Nyilvántartó Hivatalban. Ekkor kerül először kiszámításra ugyanazzal az algoritmussal a 11. szám, mint amivel később minden beíráskor újra kiszámítódik, az ellenőrzés céljából.

A Személyi szám ellenőrző számának képzési eljárása az úgynevezett **sorszám szorzatok halmozása majd 11 –es moduló** számításon alapul. Ez azt jelenti, hogy vesszük sorban az 1. – 10. számjegyeket, beszorozzuk mindegyiket a sorszámukkal, majd az így kialakult számokat összeadjuk és végül ezt az összeget elosztjuk 11 –gyel. Az így létrejött maradék lesz az Ellenőrző szám.

Az 1977. április 10. –én született, 216 –os sorszámmal rögzített férfi Személyi számának Ellenőrző száma:

Sorszámok :	1	2	3	4	5	6	7	8	9	10
A Személyi szám jegyei :	1	7	7	0	4	1	0	2	1	6
Szorzatok :	1	14	21	0	20	6	0	16	9	60

A szorzatok összege :	147
A szorzatok összege **/ 11** :	13
Az osztás maradéka :	**4**

A teljes Személyi szám tehát: **1 770410 216 4**

1.5. AZ ADATÁLLOMÁNY RENDEZETTSÉGE

Az adatokra többnyire meghatározott sorrendben – rendezettségben – van szükség. Ha rendezetten állnak rendelkezésre könnyebb keresni közöttük, gyorsabban lehet kiválasztani a szükséges adatokat. Az adatok (adatrekordok) sorrendjét mindig a feldolgozás szempontjai határozzák meg.

Pl. A tanulók adatait mindig más – más sorrendbe kell állítani akkor, ha Név szerint keresünk valakit, illetve névsort íratunk ki a nyomtatón, vagy ha az azonos Településen lakók listáját akarjuk megkapni, vagy ha az azonos Évben és Hónapban születettek adataira vagyunk kíváncsiak.

A rendezési előírás az adatrekord bármelyik mezőjére, sőt a rekordot alkotó mezők együtt értelmezett csoportjára is megadhatók.

Pl. Vezetéknév + Utónév a névsor esetén, a Település neve az egy helyen lakók listájához, Év + Hónap az azonos év-hónapban születettek feldolgozásához, Év + Hó + Nap a pontos születési sorrend megadásához.

A többszörös rendezettségi feltételt mindig úgy kell értelmezni, hogy a rendezés az 1. feltétel és ezen belül a 2. feltétel és ezen belül a 3. - és így tovább feltétel alapján történik.
Ezt úgy mondjuk, hogy **a rendezés egymásba ágyazottan történik**.

Pl. Az azonos Vezetéknevűek belső sorrendje az Utónév szerinti sorrend, az azonos Éven belül felsorolásra kerülnek a Január, majd a Február, majd a Március - és így tovább - hónapban születettek, majd a December után kerülnek felsorolásra a következő Év Januári, Februári és így tovább, születésűek.

A rendezettség további jellemzője, hogy **a rendezettség** Növekvő **(ASCENDING)** illetve Csökkenő **(DESCENDING)** sorrendű.

Pl. A névsorra a hagyományos módon A –tól haladva az ABC vége felé vagy ZS –től visszafelé haladva van-e szükség, vagy a születési adatok felsorolását a legfiatalabbtól illetve a legidősebbtől akarjuk kezdeni.

A Növekvő rendezettséget az adatbázis kezelő rendszerek **alapértelmezés**nek tekintik, azaz csak akkor kell előírni rendezettségi irányt ha az Csökkenő.

Az adatbázist alkotó adatállományok azonban a legritkább esetben – általában sosem – rendezettek, illetve ha egy szempont szerint rendezettek is egy másik szerint már biztosan nem azok.

1.6.1. A RENDEZETTSÉG ELLEN HATÓ TÉNYEZŐK

1. Amikor adatokat viszünk fel, azaz WRITE művelettel feltöltjük az adatállományt, akkor valamilyen nem számítástechnikai adathordozón tárolt (kézzel vagy géppel kitöltött űrlapok, bizonylatok, jegyzékek, listák, stb.) adatok billentyűzeten történő beírása történik. Az adatállomány egy-egy rekordját az egy-egy űrlapon feltüntetett adatok alkotják.

Pl. Ilyen például az Ideiglenes lakcím bejelentő karton, amelynek adatait államigazgatási eljárás során rögzítik a Lakcím-nyilvántartási adatbázisban. Amikor először kialakították ezt az adatbázist, akkor milliós nagyságrendű ideiglenes lakcím adatot kellet felvinni. Ha ezeket az adatokat mondjuk Lakcím sorrendben akarták volna rögzíteni, elképzelhető, hogy milyen bonyolult és sokáig tartó munka lett volna a kartonokat sorba rendezni.

2. Az adatrögzítés, mint a fenti példában is láthattuk, általában sok-sok adat felvitelét jelenti. (Egy közepes áruház is tízezres nagyságrendű árucikk-féleséggel dolgozik.) A szigorú sorrendben történő felvitel csak úgy lenne lehetséges, ha egyetlen adatrögzítő munkahelyen történne az adatok beírása, ami szintén sok időt venne igénybe. Ezért a munkát több, egy időben dolgozó számítógépen végzik el. Az egyes adatrögzítő helyek megkapják a saját – esetleg rendezett – bizonylat csomagjukat és elkezdik azokat berögzíteni. Az adatrekordok tehát egyszerre több géptől is érkeznek a közös adatállományba és ekkor már rögtön felborul a nehézen létrehozott rendezettség.

3. Az adatállomány tartalma a feltöltés után a legritkább esetben marad változatlan, az élet megy tovább, változásokat kell átvezetni az adatállományban. Az ilyen adattartalom változást az állomány aktualizálásának, **UP TO DATE** –nek nevezzük. Az adatállomány aktualizálása a WRITE, REWRITE és DELETE adatbázis műveletekkel történik.

3.1. A rendezettségre a törlés (DELETE művelet) tulajdonképpen nincs hatással, mert egyszerűen csak "kiesik" egy adatrekord a sorból.

3.2. Az új rekord felvitele, akár kötegelt (batch), akár interaktív módon (WRITE művelet) csaknem 100%-os bizonyossággal a rendezettség felborulását eredményezi, mivel az új adatrekord igen nagy valószínűséggel valahol az állomány "belsejében" kellene, hogy helyet kapjon, mert a rendezettség előírása szerint ott következne. Az ilyen "beszúrás" jellegű adatfelvitelt azonban az adatbázis kezelő rendszerek nem teszik lehetővé. (Nincs is rá szükség, mert az adatbázis kezelőket többek között éppen az ilyen feladatok megoldására találták ki !)

3.3. A, adatmódosítás (REWRITE művelet) pedig azért okozhatja a rendezettség felborulását, mert ha éppen egy olyan adatot módosítunk, amely szerint a rendezettséget meghatároztuk, akkor az adatrekordnak át kellene sorolódni az új adattartalom szerinti pozícióba. Márpedig az ilyen adatrekord mozgást sem támogatják az adatbázis kezelők. (Mert erre sincs szükség !)

1.6.2. A RENDEZETTSÉG LÉTREHOZÁSA

Az adatállományok rendezettségére, mint azt már korábban megállapítottuk, a feldolgozás során van szükség, nem utolsó sorban azért, hogy a feldolgozás gyorsabb legyen. Az adatbázis kezelő rendszerek ezt a problémát kétféleképpen oldják meg.

Az egyik módszer egy kézenfekvő megoldás, vagyis előállítják az adatállomány előírt sorrendiségű másolatát. Ez a **SORT** rendező művelet. A művelet során a memóriában vagy a háttértárolón előáll egy, az eredetivel azonos méretű és tartalmú de más rekord sorrendű új adatállomány. Az így lehetségessé vált gyorsabb feldolgozásnak azonban ára van, mégpedig az, hogy annyiszor több helyet foglal el a tulajdonképpen azonos tartalmú állomány, ahányféle rendezettségi sorrendet kialakítottunk.

A másik módszer az úgynevezett **INDEXELÉS.** Ekkor egy **INDEXTÁBLA** jön létre, amely az adatállomány duplikálása nélkül biztosítja a kívánt rendezettséget. Anélkül, hogy itt részletesen tárgyalnánk az indexelés folyamatát, a lényege ennek a módszernek az, hogy létre jön a rendezettségi előírásnak megfelelő indextábla, amely nem az eredeti adatállományt, hanem lényegében csak annak fizikai címét tartalmazza (a háttértárolón). Ezen fizikai címek azonban olyan sorrendben következnek egymás után az indextáblában, ami az előírt rendezettséget tükrözi. A kívánt rendezettségű feldolgozás során, az adatállományt és az indextáblát együtt használja az adatbázis kezelő olymódon, hogy olvassa sorban az indextábla rekordjait és megkeresi az általa kijelölt, megfelelő sorszámú adatállomány rekordot, amit fel kell dolgoznia.

Pl.

A rekord fizikai címe	A rekord tartalma és tárolási sorrendje		
	Típus	Évj.	Szín
1	Volkswagen	1999	Metál
2	Audi	1980	Piros
3	Skoda	2000	Fekete
4	Opel	1993	Zöld
5	Fiat	1997	Piros

Az INDEXTÁBLA tartalma és rendezett sorrendje	
A rekord fizikai címe: POINTER*	Típus (a rendezési szempont)
2	Audi
5	Fiat
4	Opel
3	Skoda
1	Volkswagen

* A POINTER jelentése: **MUTATÓ**

Az így létrehozott rendezettség szintén helyet foglal de korántsem annyit, mint a SORT művelet eredménye. Gondoljuk meg, hogy nem csak az autó márkáját év gyártási évét, hanem sok más egyéb adatát is tároljuk a fenti példa állományban.

A rendezettség vizsgálata illetve előállítása során lényeges kérdés, hogy megengedhető-e a rendezésben résztvevő adat többszörözöttsége. Vagyis mi a teendő akkor, ha a rendezés során kettő (vagy több) azonos tartalmú adatot kell sorrendbe állítani. Ez a fenti példa esetén azt jelenti, hogy kettő (vagy több) azonos típusú autó is szerepel a listában.

Az egyszerű rendezés (SORT) esetében nincs különösebb teendő, az azonos rekordok az eredeti sorrend szerint egymás után íródnak.

Az indexelés esetén azonban probléma adódhat abból, hogy több azonos rendezési feltételt tartalmazó rekord is van. Kérdés tehát, hogy a fenti példa esetében hogyan kell eljárni akkor, ha netán kettő (vagy több) Audi is szerepel a listán. Ezen probléma megoldása és a megfelelő döntés meghozatala a rendszert kialakító szakember (a Rendszerszervező) feladata, amikor megtervezi az adatbázist. Dönthet úgy, hogy megengedhető az adat többszörözöttség, azaz egyszerű indexet hoz létre, de úgy is, hogy nem, azaz egyedi **UNIQUE INDEX** –et hoz létre. A fenti példa esetében ez úgy érhető el, hogy a rendezettségbe bevonja az autótípus mellé az évjáratot is és azzal a kikötéssel él, hogy nem lehet a listában két egyező évjáratú, azonos típusú autó. (Ha ez sem teljesíthető, akkor további feladatai vannak, amelyet később részletesen meg fogunk ismerni.)

1. 6. AZ ADATÁLLOMÁNYOKKAL VÉGEZHETŐ MŰVELETEK

1. **Az adatállomány olvasása** anélkül, hogy tartalmában változás történne. Ezt az adatbázis művelet nevezzük **READ** műveletnek.
2. **Új adatrekordok felvitele** az adatállományba, amelyet **WRITE** műveletnek nevezünk.
3. **Létező adatrekord tartalmának módosítása**, vagyis egy vagy több mező tartalmának megváltoztatása, azaz felülírása. Ezt az adatbázis műveletet **REWRITE** műveletnek hívjuk.
4. **Adatrekordok törlése** az adatállományból. Ezt a műveletet **DELETE** műveletnek nevezzük.

Megjegyzés: a törlés művelete általában kétlépcsős, előbb logikai törlés (kijelölés törlésre) majd fizikai törlés történik.

2. ADATBÁZIS MODELLEK

Az adatbázis modellek, a vizsgált rendszer elemeire nézve leírják azok fontos tulajdonságait, összefüggéseit és tartalmukat, valamint meghatározzák felhasználásuk körülményeit.

Az adatbázisok LOGIKAI adattárolási módszereit értjük Adatbázis modell alatt.

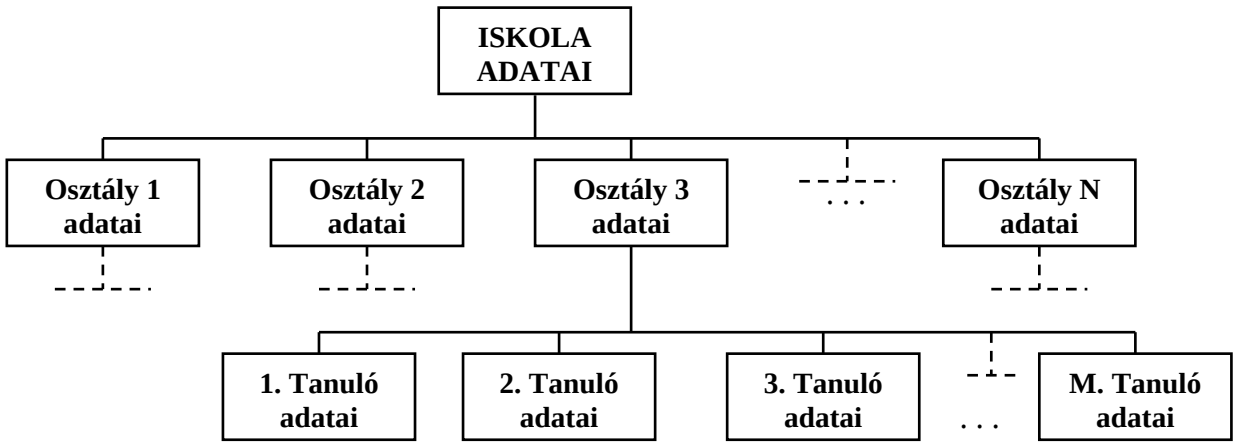
Az egyes modellek, elsősorban az adatkapcsolatok jellegében térnek el.

2.1. HIERARCHIKUS adatbázis modell

Ez az ABKR -ek kialakulása során először alkalmazott modell. Az 1960 –as évek vége felé kezdték alkalmazni. Nevéből is következik, hogy ez egy alá - fölérendeltségi viszonyon alapuló adatszerkezet. (tulajdonos - tulajdon). A rekordok tartalmazzák azokat az adatokat, amelyek el kívánunk tárolni és azt a rekord azonosítót, amelyhez mint "tulajdon" kapcsolható.

Gyakorlatilag egy **FA struktúráról** van szó, ahol a GYÖKÉR -ből (angolul ROOT) kiindulva elágazások, más néven CSOMÓPONT-ok sorozatán keresztül jutunk el a tovább már nem elágazó csúcshoz..

Pl. Hierarchikus szerkezetek a Családfa, Főnök - Beosztottak, Iskola - Osztályok - Tanulók adatbázisok.



Az ISKOLA / Osztály N / M. Tanuló feliratú téglalapok mindegyike, az adott objektumhoz tartozó adatsorozatot szimbolizálja (Például a Tanulók adatai: Név; Lakcím; Születési dátum; Kollégista; stb. és az ***Osztály 3*** –ba jár illetve az Osztályok adatai: Osztály neve; Létszáma; Ebből leány; Kollégista; Osztályfőnök neve, stb. és az ***ISKOLA*** azonosítója).

A Tanulók és az Osztályok adatsorában tehát fel kell tüntetni az adatsor ***Felső kapcsolatát***, amit **TULAJDONOS** –nak (angolul **OWNER**) nevezünk.

Az Osztály – Tanuló illetve az ISKOLA – Osztály kapcsolatban a Tanulók illetve az Osztályok ***Alsó kapcsolatként*** szerepelnek, amit **TAG** –nak (angolul **MEMBER**) nevezünk.

Az Osztály abban a különleges helyzetben van, hogy egyszerre OWNER és MEMBER is.
Az ISKOLA, mint az adatbázis GYÖKÉR eleme csak OWNER, a Tanulók viszont csak MEMBER –ek.

Az állományok tartalma:

ISKOLA : IKOD - az ISKOLA egyedi azonosító kódja,
továbbá, az iskola egyéb adatai . . .. ;

Osztály N : OKOD - az osztály egyedi azonosító kódja,
IKOD - a (tulajdonos) ISKOLA azonosító kódja,
továbbá, az Osztály egyéb adatai . . . . . ;

M Tanuló : TKOD - a TANULÓ egyedi azonosító kódja,
OKOD - a (tulajdonos) OSZTÁLY azonosító kódja,
továbbá, a tanuló egyéb adatai . . . . . ;

A Hierarchikus adatmodell jellemzői:

- csak **1 : N típusú kapcsolat** lehetséges az AB –ban
- a Fa csomópontja és levelei az adatrekordok
- alá-fölérendeltségi viszony van a rekordok között
- az alárendelt (MEMBER) rekord csakis egyetlen rekordhoz tartozhat, a Tulajdonosához
- a fölérendelt (OWNER) rekordhoz egy vagy több rekord tartozhat
- egy közös szempont (pl. a tanulók lakhelye) szerinti lekérdezése vagy csoportosítása, mindig a gyökérből kiindulva és az összes csomóponton áthaladva lehetséges.

2.2. HÁLÓS AB modell

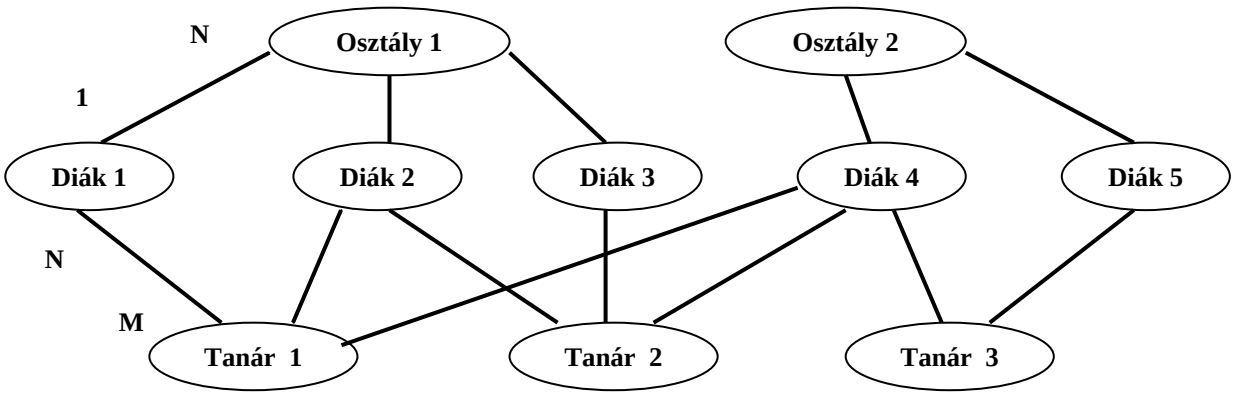
A szoftver és nem utolsó sorban hardver fejlődésének köszönhetően terjedhetett el az 1960 –as évek végétől a Hálós adatbázis modell. Az adatkapcsolatok mellérendelt jellegűek (nincs tulajdonos), lehetséges az N : M és nyilván lehetséges ennek különleges esete az 1 : N típusú kapcsolat is.

A modell egyik képviselője a **CODASYL** úgynevezett irányított gráf típusú adatbázis modell. Ez a név a **CON**ference on **DA**ta **SY**stem **L**anguage rövidítése, ahol 1971-ben először ismertették ezt az adatbázis modellt és a adatbázis feldolgozás követelményeit. Az irányított gráf azt jelenti, hogy egy külön állományban kerülnek rögzítésre az adatállományok rekordjai közti kapcsolatok, amelyek logikai összefüggés alapján tulajdonos rekord → tagrekord irányított kapcsolatokat határoznak meg. Az adatállomány rekordjait **rekordtípusnak** a kapcsolati rekordot **settípusnak** nevezük.

A másik elterjedt modell az **IBM** gépekre készített **IDMS** (**I**ntegrated **D**atabase **M**anagement **S**ystem) adatbázis kezelő. Ebben már megjelennek az adatbázis kezelő rendszerek nyelvi elkülönülései (lásd DDL, QL)

Ilyenek: 1 : N Osztály - Tanulók adatainak kapcsolata

N : M Diákok - Tanárok adatainak kapcsolata



A csomópontok az ADATOK, az élek a KAPCSOLATOK.
Az adatok között itt nem szerepelnek a kapcsolatok, hanem külön kapcsolat leíró állományt

kapcsolat MÁTRIX –ot alkalmaznak

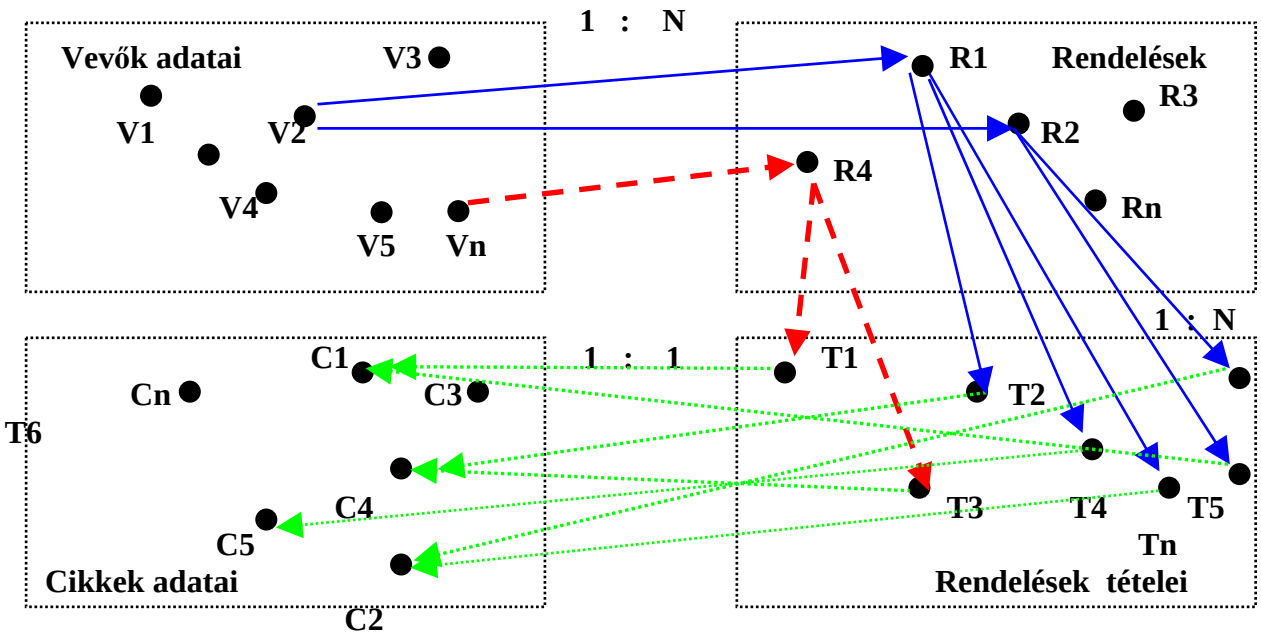
Az Osztály - Diák kapcsolat mátrixa 1 – N a kapcsolat, mert a diák sorában csak egy, az Osztály oszlopában pedig több érvényes kapcsolat jelző (1) van.

O - D	Osztály 1	Osztály 2
Diák 1	**1**	**0**
Diák 2	**1**	**0**
Diák 3	**1**	**0**
Diák 4	**0**	**1**
Diák 5	**0**	**1**

A Diák - Tanár kapcsolat mátrix viszont minden oszlopában és sorában több kapcsoló jelzőt is tartalmaz, tehát N – M kapcsolat áll fenn.

D - T	Diák 1	Diák 2	Diák 3	Diák 4	Diák 5
Tanár 1	**1**	**1**	**0**	**1**	**0**
Tanár 2	**0**	**1**	**1**	**1**	**0**
Tanár 3	**0**	**0**	**0**	**1**	**1**

A hálós AB modell esetenként igen hatékony, gyors az adatelérés, mert a kapcsolatok fixen beépítettek az AB leírásába. Ugyanakkor ez rossz is lehet, amikor módosítani kell az AB szerkezetét, mert mindent újra kell definiálni.
Egy Rendelés nyilvántartó AB modellje HÁLÓS szerkezetben:



A "Rendelés Nyilvántartás" adatbázist a "Vevők adatai", "Cikkek adatai", "Rendelések" és a "Rendelések tételei" nevű adatállományok alkotják. A fekete körök jelentése az egyes állományokban, egy - egy konkrét elem (egy Vevő, egy Cikk, stb.) adatait jelentik.

Vevők adatai: { Vevő azonosító, Vevő neve, Címe, Telefonszáma, stb. }

Cikkek adatai: { Cikkszám, Cikk elnevezése, Ára, Csomagolása, Szavatossági ideje, stb. }

Rendelések: { Rendelés azonosító, Melyik vevő rendelése (Vevő azonosító), A megrendelés dátuma, A szállítás határideje, Hány tétel szerepel a rendelésen, A megrendelés összege, stb. }

Rendelések tételei: { Melyik megrendeléshez tartozik (Rendelés azonosító), Melyik Cikkre vonatkozik a rendelés tétel (Cikkszám), Mennyit, rendelt, stb. }

Az állomány kapcsolatok: [Vevők adatai **1 : N** Rendelések], a Vevő több Rendelést is feladhat

[Rendelések **1 : N** Rendelés tételei] egy Rendelésen több Tétel szerepelhet

[Rendelések tételei **1 : 1** Cikkek adatai] egy Rendelés tétel csakis egy Cikket tartalmazhat

Létezik még egy "rejtett" kapcsolat is, a Vevők és Cikkek között, amely N : M –es. Ezt a kapcsolatot szükségtelen mátrixban rögzíteni, mert a nyilakon visszafelé haladva megállapítható, hogy melyik vevő milyen cikkeket rendelt, illetve egy adott cikket melyik vevők rendelték.

A V2 –es Vevő, R2 –es Megrendelésén 3 tétel van (T1, T4 és Tn), amelyek a C4, C5 és C2 Cikkekre vonatkoznak. A V2 –es Vevő megrendelése az R2 –es is, amely a T5(C1) és T6(C2) tételeket tartalmazza.

A Vn Vevőnek csak egy megrendelése van R4 azonosítóval, amely a T1 és T2 tételeket tartalmazza és a C1 illetve C4 Cikkekre vonatkozik.

Fontos észrevenni, hogy egy Rendelés tétel csakis egy Cikkel lehet kapcsolatban, de ugyanazt a Cikket egy másik Rendelés tétel is tartalmazhatja.

3. A RELÁCIÓS AB modell

A legelterjedtebb AB modell, a matematikai **halmazelméleten alapul**. **E. F. CODD dolgozta ki az elméletét, 1971 -ben.** Elméletének továbbfejlesztésében **Halassy Béla** alkotott maradandót.

A reláció fogalma: adathalmazok DESCARTES szorzata által létrejött táblázat.

A DESCARTES szorzat két halmaz egyesítése olyan módon, hogy az egyik halmaz minden egyes elemé-hez hozzárendeljük a másik halmaz minden elemét.

Reláció például a lehetséges összes vezetéknév és a magyar utónevek összerendelésével keletkező személynév halmaz, amely biztosan tartalmazza bármelyik magyar ember nevét.

pl. a BETÜK { A, B, C, D } és a SZÁMOK { 1, 2, 3, 4, 5, 6 } halmazok DESCARTES szorzata

{ A1, A2, A3, A4, A5, A6,
B1, B2, B3, B4, B5, B6,
C1, C2, C3, C4, C5, C6,
D1, D2, D3, D4, D5, D6 }

Fontos ! A táblázat nem tartalmazhat két (vagy több) azonos tartalmú sort,
a rekordok sorrendje a táblázaton belül tetszőleges,
az adatok, vagyis a táblázat oszlopainak sorrendje szintén tetszőleges lehet.

A RELÁCIÓS AB tehát egymással meghatározott kapcsolatban álló, TÁBLÁZAT -ok halmaza.
Lényege, hogy **az adatok kapcsolata csak logikai szinten kerül meghatározásra**,
azaz nincs fizikai kapcsolat leírás sem az adatállományokban sem külön állományban (kapcsoló mátrix).

Az adattárolás, táblázatos (sor - oszlop) formában történik, a logikai adatkapcsolat a táblák között, kapcsoló adatelemek, **kulcsok** segítségével biztosítható.

3.1. A RELÁCIÓS AB ALAPFOGALMAI

EGYED : Minden létező, amiről adatokat tárolhatunk;

EGYED TÍPUS : maga a TÁBLÁZAT - az a konkrét Egyed, aminek a leíró adatait tároljuk
(pl. autók, az osztály tanulói, árucikkek);

EGYED ELŐFORDULÁS : a Táblázat SOR -a (REKORD másként TUPLE);

TULAJDONSÁG TÍPUS : a Táblázat OSZLOP -a (más néven ATTRIBÚTUM);

- az Egyed különböző, adatainak (tulajdonságainak) neve (NEM a konkrét értéke !)
(pl. RENDSZÁM, TÍPUS, SZÍN, ÁR, stb.);

- egyes szakkönyvek ezt nevezik MEZŐ –nek !
- A tulajdonság lehet: leíró, azonosító és kapcsolat meghatározó.

KULCS : a táblák kapcsolatának meghatározója (Lásd később részletesen)

TULAJDONSÁG ELŐFORDULÁS : a Táblázat Oszlop/Sor metszete (MEZŐ más néven FIELD)

- a Tulajdonság Típusok konkrét értékei, adatok
  (pl. GFA-623, VW GOLF, KÉK, 3,6 MFt, stb.).

INDEXELÉS: a táblázatok adatsorrendjének meghatározása.

A RELÁCIÓ FOKA: a táblázat attribútumainak (oszlopainak) száma.

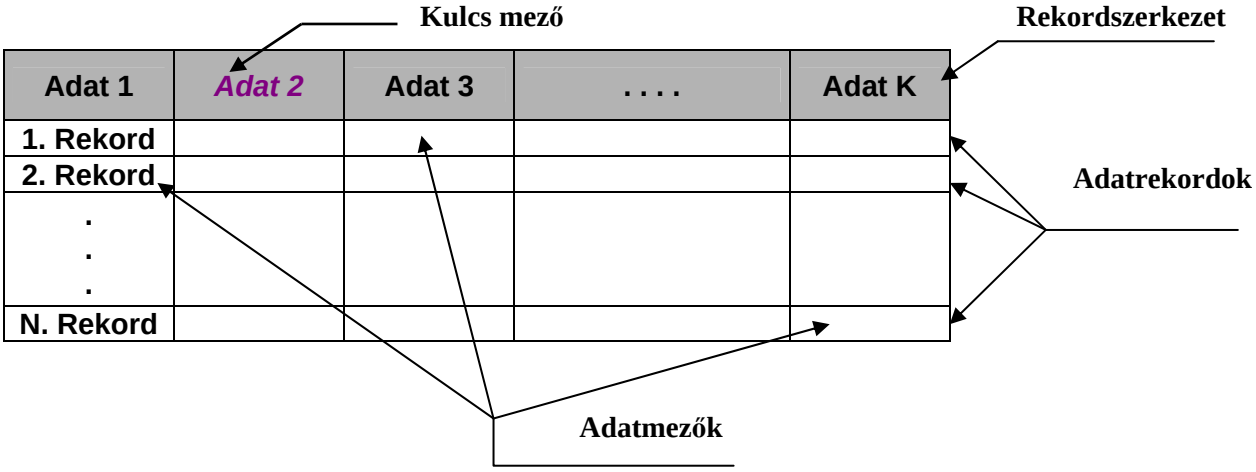
Megjegyzés: a szakkönyvek ezeket az alapfogalmakat eltérő megnevezésekkel is jelölhetik, amelyeket mindig a fogalom tényleges jelentésével kell összevetni és alkalmazni.

Az adat sorrendekre vonatkozó megállapítás ezek után a következő:

Az egyed előfordulások és az attribútumok sorrendje indifferens.

3.2. A RELÁCIÓS AB ADATÁLLOMÁNY SZERKEZETE

Az Adatállomány egy **TÁBLÁZAT,** amely tehát egy előre megtervezett adatszerkezet szerinti adatsorokból, a Rekordokból áll.



3.3. A REKORDSZERKEZET alatt az adatok tulajdonságainak és sorrendjének rögzítését értjük.

Tervezése során határozzuk meg, hogy milyen adatokat, milyen formában és milyen egyéb előírások szerint tárolunk az adatállományban.

Meg kell adni: **Adat nevét**

Típusát *

Hosszát (szám esetén azt is, hogy hány tizedes pontosan tároljuk)

Egyéb előírásokat **

* A megadható adattípusok egyes adatbázis kezelő rendszerek esetében eltérőek lehetnek. Az általános adattípusok azonban mindegyikben érvényesek és ezeknek a típusoknak a kibővített változatait alkalmazzák.

** Egyéb előírás lehet: kötelező-e a mezőt kitölteni, vagy maradhat üresen is,
valamilyen alapértelmezett érték hozzárendelése,
szeretnénk-e rendezési szempontként alkalmazni az adatot, stb.

3.4. Az adatbázisokban általánosan használt adattípusok

KARAKTERES - betűk, számok, jelek

NUMERIKUS - egész és tizedes számok (esetleg normál alakban)

DÁTUM - általában formázható (pl. ÉÉÉÉ/HH/NN vagy HH/NN/ÉÉ)

LOGIKAI - tartalma csak IGEN / NEM illetve IGAZ / HAMIS (Y/N ill. T/F) lehet

KÖTETLEN (RAW) vagy BINÁRIS - alkalmas nagyméretű szövegek, képek, térképek tetszőleges állományok bináris formájú tárolására

3.5. A NULL ÉRTÉK fogalma

Az adatdefiníció során a név, hossz típus mellett tartalmi előírások is hozzárendelhetők az adott tulajdonságtípushoz. (dátum intervallum, min-max érték, stb.)

Jelentősége abban áll, hogy lekérdezések esetén az „üres” tartalom is jelenthet valamit, de ha azért üres mert nem tudtunk még értéket adni az adatnak és ez kizáró ok arra , hogy lekérdezésben szerepeljen, akkor a típusát „null értéknek” kell előírni.

A logikai null érték definíciója

VAGY relációban:

	T	**F**	**?**
T	T	T	T
F	T	F	?
?	T	?	?

ÉS relációban:

	T	**F**	**?**
T	T	F	?
F	F	F	F
?	?	F	?

3.5. AZ ADATBÁZIS KEZELŐ RENDSZER

Az adatbázisokat feldolgozó programrendszereket **A**dat**B**ázis **K**ezelő **R**endszereknek röviden ABKR nevezzük. Angol megnevezése a **D**ata**B**ase **M**anagement **S**ystem (DBMS)

Az ABKR által hozható létre, tartható karban, lekérdezhető, egyszóval üzemeltethető illetve védhető az adatbázis.

Az ABKR elemei: adatdefiníciós nyelv (**D**ata **D**efinition **L**anguage),
adatkezelő nyelv (**D**ata **M**anipulation **L**anguage) ,
lekérdező nyelv (**Q**uery **L**anguage) - lásd **SQL** ,
adat és üzemeltetési biztonsági utasítások.

3.7. ADAT LEKÉRDEZÉS AZ ADATBÁZISBÓL

Az adatbázis feldolgozásának lényegi kérdése a rendezettségen túl, hogy többnyire nem a teljes adatállomány tartalomára van szüksége a felhasználónak, hanem csak valamilyen feltételnek megfelelő rekordjaira illetve a rekordok szűkített adattartalmára. Az is gyakori igény, hogy megjelenítendő az adatokat több állományból kell "összeszerkeszteni".

Ezeket a feladatokat megvalósító adatbázis műveletet nevezzük **LEKÉRDEZÉS** –nek (angolul **QUERY**). Az adatbázis kezelők a lekérdezés tartalmát meghatározó "előírásokat" **NÉZET** –nek hívják
(angolul **VIEW**). A nézetek létrehozásában két fontos funkció szerepel, egyrészt az előírt feltételnek megfelelő rekordok leválogatása, amely műveletet **SZŰRÉS** –nek (angolul **SELECTION**) illetve a szükséges rekord tartalom megjelenítése, amely műveletet **VETÍTÉS** –nek (angolul **PROJECTION**) nevezünk. További adatbázis műveletek még az adatállományok **egyesítése**, **különbség** és **metszet képzés** valamint a **Descartes szorzat** előállítás.

Az adatbázis kezelők általában a Lekérdezéseknek két fő típusát különböztetik meg:

- az egyszerű **Látvány lekérdezést**, amely az adatokat csak megjeleníti a kívánt szerkezetben,
- és a **Módosító lekérdezést,** amely segítségével például csoportos adat módosítást vagy rekord(ok) törlését hajthatjuk végre.

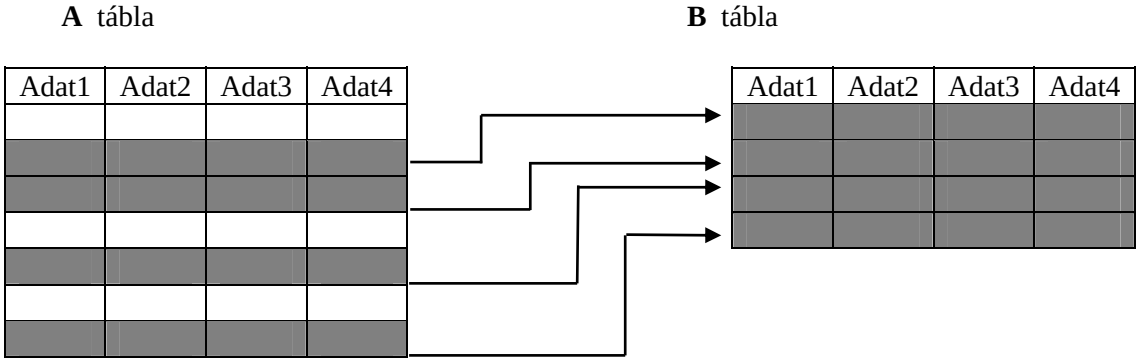
Fontos kihangsúlyozni, hogy a létrehozott lekérdezések megjelenési formája (amikor dolgozunk vele vagy megjelenítjük a képernyőn) ugyanolyan, mintha egy adatállományt dolgoznánk fel. A lényegi különbség az, hogy nem adatok "tárolódnak" a lekérdezésben meghatározott szerkezetben és formátumban, hanem csak **azok az utasítások kerülnek rögzítésre az adott nevű lekérdezés állományban, amelyek a szerkezetet és formátumot meghatározzák**.

Pl. Az autókat tartalmazó listából csak az 1998. után gyártott autók érdekelnek és a rekordban szereplő sok egyéb adat közül csak a henger űrtartalom, a szín és persze az autó rendszáma.
Az is egy jogos igény, hogy az autók és a tulajdonosaik adatait (amely adatok egy másik állományban kerültek eltárolásra) egy listán láthassuk.

Az adatbázis feldolgozásának jelentős része tehát a lekérdezés, ezért szabványosított lekérdező nyelv(ek) is forgalomba kerültek. Ezek legnagyobb körben alkalmazott képviselője az **SQL**, amely illeszkedve a legtöbb adatbázis kezelő rendszerhez, biztosítja azok egységes lekérdezését. Jelenleg a világban az 1992 –es megállapodás szerinti SQL '92 szabvány van érvényben.
SQL - Structured **Q**uery **L**anguage (Strukturált Lekérdező Nyelv)

3.8. A relációs algebra műveletei

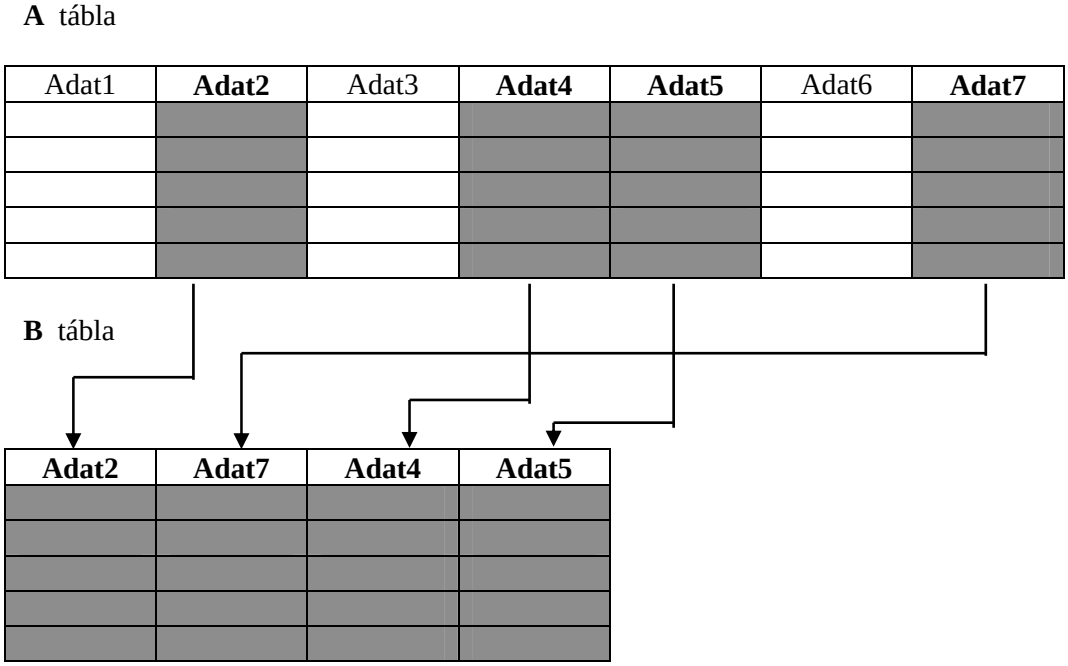
1. **Szelekció** : (**szűrés**) - adott feltételnek megfelelő egyed előfordulások (rekordok) kiválogatása és megjelenítése az eredmény táblában.
Nevezik még **Korlátozás**nak illetve **Horizontális megszorítás**nak is.



Azok a rekordok kerülnek át az **A** (forrás) tábla rekordjai közül a **B** (eredmény) táblába, amelyek egy adott feltételnek megfelelnek. (például egyik attribútumára vonatkozó reláció IGAZ értéke esetén),

pl. Azoknak a személyeknek a megjelenítése egy eredmény listában, akik érvényes az a feltétel, hogy 1980. után Móron születtek.

2. **Projekció** : (**vetítés**) - a tábla meghatározott Attribútumainak (oszlopainak) megjelenítése eredmény táblában. Lehetséges az oszlopok sorrendjének megváltoztatása is.
Nevezik még **Vertikális megszorítás**nak is.



A **B** (eredmény) tábla egyrészt szűkített adattartalmú, másrészt eltérő sorrendben tartalmazza az adatokat, mint az **A** (forrás) tábla.

pl. A személyek összes adata közül csak a Személyi szám, a Születési hely és Idő kiíratása.

3. **Descartes szorzat** : (**teljes szorzat**) - a relációk teljes szorzata, két (vagy több) tábla sorainak teljes variációs megjelenítése. Nevezik még **Direkt szorzat**nak is.

A tábla

SZ1	SZ2	SZ3
1		
2		
3		

B tábla

B1	B2	B3	B4
X			
Y			
Z			

C tábla

SZ1	SZ2	SZ3	B1	B2	B3	B4
1			X			
1			Y			
1			Z			
2			X			
2			Y			
2			Z			
3			X			
3			Y			
3			Z			

A **C** (eredmény) tábla az **A** és **B** táblák DESCARTES szorzatának eredménye.

pl. Lásd a korábbi példát, (a BETŰK és SZÁMOK nevű állományok) ahol a betűk és számok valamilyen konkrét adatot jelentenek.

4. **Join** : (**összekapcsolás**) - két (vagy több) tábla sorainak közös megjelenítése Attribútum értékre előírt feltétel alapján. (Vagyis az 1. tábla rekordjához annyiszor kapcsoljuk a 2. tábla rekordjait, ahány esetben az előírt feltétel Igaz értékű.)

Leggyakoribb esete két tábla azonos nevű Attribútumának egyezőség feltétele alapján történő összekapcsolás, amelyet **Természetes összekapcsolás** -nak nevezünk. Ebben az összekapcsolásban azok az 1. tábla rekordok + 2. tábla rekordok alkotják az eredmény táblát, amelyekben a megadott Attribútum tartalma azonos. A táblák azon rekordjai nem szerepelnek az eredmény táblában, amelyekre nem teljesül a feltétel.

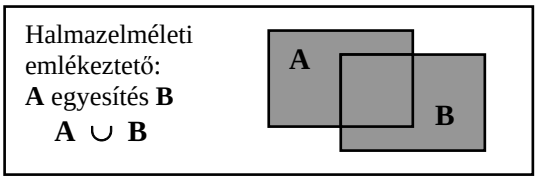
Előírható, hogy az 1. vagy a 2. tábla azon rekordjai is kiírásra kerüljenek, amelyek nem felelnek meg az egyezőség feltételének. Ebben az esetben **Külső összekapcsolásról** –ról beszélünk.

pl. A PARTNER { *Vevőkód*, Vevőnév, Vevőcím, …} tartalmú állomány és a partnerek vásárlásait tartalmazó FORGALOM { Cikkszám, *Vevőkód*, Mennyiség, Forint, … } állomány rekordjainak Természetes összekapcsolása a Vevőkód alapján. Eredménye egy olyan tábla, amely Vevőnként külön csoportosítva tartalmazza, hogy milyen Cikket (mennyit, mennyiért) vásároltak. Az eredmény táblában csak akkor szerepelnek azok a Vevők akik még nem vásároltak semmit, ha Külső összekapcsolást
alkalmazunk.

3.9. Halmaz műveletek

Fontos ! A halmaz műveletek csak azonos szerkezetű "kiinduló" táblázatok között hajthatók végre. Ez azt jelenti, hogy a feldolgozandó két (vagy több) táblázat **azonos számú attribútum**ot tartalmazhat, továbbá az azonos sorszámú (1. 2. 3. …) attribútumok **azonos típusúak.** Ezek a feltételek nem azt jelentik, hogy a forrástáblák (amelyekből az eredményt képezzük) azonos szerkezetűek. A "kiinduló" táblázatokból projekcióval kell képezni az azonos szerkezetű forrás táblákat. A halmaz művelet természetesen szűrt táblázatokra is értelmezett.

1. **Egyesítés** - két (vagy több) tábla kiválasztott tartalmának egymás után írása.
   Külön utasítás nélkül az azonos tartalmú sorok csak egyszer szerepelnek !
   SQL operátora az **UNIO**



Az **UNIÓ** művelete tehát két (vagy több azonos szerkezetű!) táblázat egymás után írását jelenti, miután projekcióval azonos szerkezetet alakítottunk ki és szűrtük a forrás táblákat. Amennyiben a táblázatokban a projekció után azonos tartalmú rekordok is előfordulnak, akkor csak az egyiket írja fel az eredmény táblába (amennyiben nem írjuk elő az egyező rekordok kiíratását is). Az eredmény tábla Attribútumainak elnevezése eltérő is lehet a forrás táblák neveitől.

A tábla

Adat 1	Adat 2	Adat 3
Aaaa		Bbbb
Dddd		**Cccc**
Eeee		Ffff

B tábla

Adat X	Adat Y	Adat V	Adat Z
	Cccc	Dddd	
	Gggg	Hhhh	
	Kkkk	Llll	

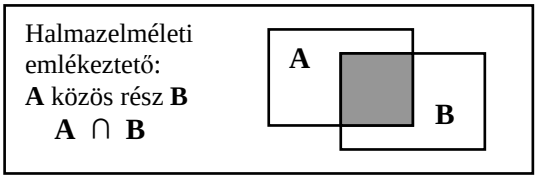
C tábla

3Y Adat	V 1 Adat
Bbbb	Aaaa
Cccc	**Dddd**
Ffff	Eeee
Gggg	Hhhh
Kkkk	Llll

Az A és B a "kiinduló" táblák. Az A tábla projekcióval létrehozott forrás táblája az { Adat 3, Adat 1 } Attribútum sorrendű tábla, a B tábla forrás táblája pedig az { Adat Y, Adat V } sorrendű tábla. Az A tábla 1. 2. és 4. rekordjai, valamint a B tábla mindhárom rekordja megfelel a megadott szűrő feltételnek, tehát szerepelhet az eredmény táblában. A két tábla egy-egy rekordja (az A 2. és a B 1. rekord) azonban azonos tartalmú, így közülük csak egy szerepel az eredmény táblában, mert nem kell az ismétlést. Összesen tehát 5 rekord szerepelhet a C eredmény táblában és az Attribútum nevek { 3Y Adat, V1 Adat }.

Pl. A Vevő (A tábla) és Szállító (B tábla) üzleti partnerek megnevezése és postai címe (C tábla).

2. **Metszet** - két (vagy több) tábla, azonos tartalmú rekordjainak kiírása eredmény táblába.
A metszet szigorúan csak az egyik rekordot írja ki.
SQL operátora az **INTERSECT**



Az **INTERSECT** a "kiinduló" táblákból, a projekcióval (azonos szerkezetű és adat sorrendű) és szűrés-sel kialakított forrás táblák azon rekordjait jelenítik meg, amelyek mindkét táblában megtalálhatók. A megjelenítés azonban csak az egyik rekordot tartalmazza.

A tábla

Adat 1	Adat 2	Adat 3
Mmm	Nnnn	
Pppp	Rrrr	

B tábla

Adat X	Adat Y	Adat V	Adat Z
Rrrr			Pppp
Nnnn			Mmm

C tábla

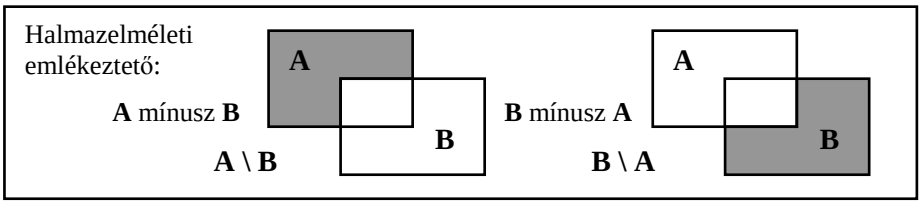
Közös 1	Közös 2
Nnnn	Mmm
Rrrr	Pppp

Az A és B a "kiinduló" táblák. Az A tábla projekcióval létrehozott forrás táblája az { Adat 2, Adat 1 } Attribútum sorrendű tábla, a B tábla forrás táblája az { Adat X, Adat Z } sorrendű tábla. Az A tábla 1. rekordja azonos tartalmú a B tábla 3. rekordjával, valamint az A tábla 4. rekordja azonos tartalmú a B tábla 2. rekordjával. Ebben az esetben nem adtunk meg szűrő feltételt egyik táblára sem, tehát minden rekordjuk részt vesz a metszet képzésben. Összesen tehát 2 rekord szerepelhet a C eredmény táblában és az Attribútum nevek { Közös 1, Közös 2}.

Pl. Azoknak az üzleti partnereknek (Vevők és Szállítók) a listája, amelyektől vásároltunk is és szállítottunk is nekik

.

3. **Különbség** - csakis 2 táblára alkalmazható művelet. Fontos a táblák sorrendje a műveletben.
Az eredmény tábla csak azokat az 1. táblázat rekordokat tartalmazza, amelyeknek nincs azonos tartalmú párjuk a 2. táblázatban.
SQL operátora az **EXCEPT** (néhány SQL nyelvben a **MINUS**)



Az **EXCEPT** művelete a "kiinduló" táblákból azon rekordokat fogja eredményezni, amelyek a projekció és szelekció művelete után az első táblában léteznek, de nem léteznek a második táblában.

A tábla

Adat 1	Adat 2	Adat 3
	Aaaa	Bbbb
	Hhhh	**Jjjj**
	Cccc	Dddd
	Eeee	Gggg

B tábla

Adat X	Adat Y	Adat V	Adat Z
	Hhhh		**Jjjj**
	Nnnn		Mmm
	Pppp		Rrrr

C tábla = **A \ B**

AB3Z	AB2Y
Aaaa	Bbbb
Cccc	Dddd
Eeee	Gggg

D tábla = **B \ A**

BA3Z	BA2Y
Nnnn	Mmm
Pppp	Rrrr

Az A és B a "kiinduló" táblák. Az A tábla projekcióval létrehozott forrás táblája az { Adat 3, Adat 2 } Attribútum sorrendű tábla, a B tábla forrás táblája az { Adat Z, Adat Y } sorrendű tábla. Az A tábla 2. rekordja azonos tartalmú a B tábla 1. rekordjával. Ebben az esetben sem adtunk meg szűrő feltételt egyik táblára sem, tehát minden rekordjuk részt vesz a különbség képzésben.
Az A mínusz B eredménye pedig az A tábla 1. 3. és 4. rekordja, Attribútum nevei { AB3Z, AB2Y }
A B mínusz A eredménye tehát a B tábla 2. és 3. rekordja, Attribútum nevei { BA3Z, BA2Y }

Pl. Írassuk ki azokat a partnereket, amelyek csak vevők, illetve azokat, akik csak Szállítók.

4. RELÁCIÓS AB TERVEZÉS

Előzménye: a Rendszerszervezésben tanultak (helyzetfelmérés, elemzés, megvalósíthatósági tanulmány)

Az AB tervezés fázisai: LOGIKAI tervezés
FIZIKAI tervezés

Adatelemzés:

Az adatelemzés feltárja a vizsgált rendszerben használt és/vagy használandó adatok:

1. Összefüggéseit,
2. Típusát,
3. Jellegzetes vagy engedélyezett értékét,

4. Keletkezési helyét,
5. Felhasználási helyét,
6. Közvetítését,
7. A keletkezés és felhasználás gyakoriságát, eloszlását.

Míg az 1. - 3. pontok inkább az AB logikai tervezéséhez fontosak, addig a többi az üzemeltetési folyamat (fizikai tervezés) megszervezésében játszik elsősorban szerepet.

LOGIKAI tervezés :

- Az EGYED és KAPCSOLATI modell meghatározása,
- Táblázat NORMALIZÁLÁS,

FIZIKAI tervezés :
(ABKR függő tevékenység,
SQL alkalmazás és
programozás)

- A Táblázat fizikai létrehozása,
- Nézetek és indexek kialakítása
- Input / Output elemek tervezése
- A működési feltételek tervezése

4.1. Az AB Logikai tervezése

Logikai tervezés: az AB szerkezetére (táblák és kapcsolatok) vonatkozó, a tényleges megvalósítást előkészítő tervező munka. Összefüggés elemzés.

A Rendszerszervezési munka során eddig a fázisig tisztázódott, hogy milyen adatokat, milyen feldolgozó környezetben, milyen cél érdekében, milyen módszerrel lehetséges feldolgozni.

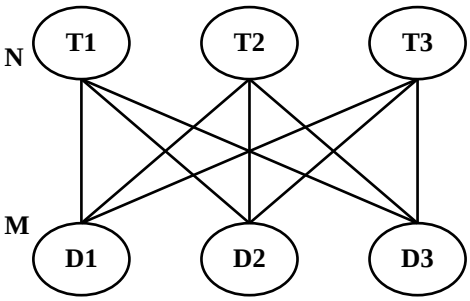
Most következik az adatok feldolgozási logika szerinti csoportosítása, összerendelése, a köztük levő kapcsolatok meghatározása. Ez a folyamat a LOGIKAI TERVEZÉS.

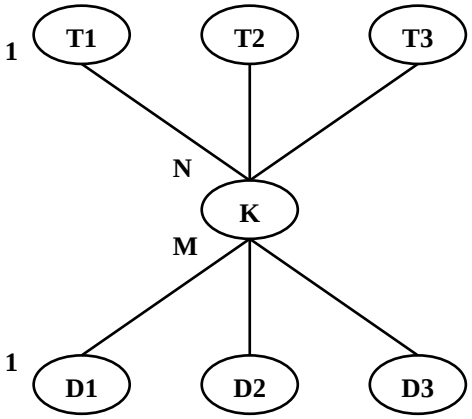
Ennek első fázisa az adatok csoportosítása logikai összetartozásuk alapján, ez az EGYED TÍPUS képzés majd ezen csoportok kapcsolatainak, a FUNKCIONÁLIS FÜGGŐSÉG -ek meghatározása.

Ezt követi az Adatelemzés további két pontja a Tulajdonságtípusok vagyis az adatok típusának (numerikus, egész. stb.) és szükség esetén azok tartalmának (nem lehet üres, mai dátumnál csak kisebb lehet, stb.) meghatározása.

A Funkcionális függés meghatározása:

- A kialakított adatcsoportok (egyed típusok) kapcsolatainak meghatározása
- A kapcsolatokat biztosító elemek meghatározása (ezek a kulcsok)
- Az N : M típusú kapcsolatok felbontása N : 1 és 1 : M típussá





Diákok tábla
1. Diák
2. Diák
3. Diák
4. Diák
5. Diák

Kapcsolat tábla	
1. Diák	2.Tanár
1. Diák	3.Tanár
1. Diák	1.Tanár
2. Diák	1.Tanár
2. Diák	3.Tanár
3. Diák	1.Tanár
4. Dák	1.Tanár
5. Diák	2.Tanár
5. Diák	3.Tanár

Tanárok tábla
1. Tanár
2. Tanár
3. Tanár

4.2. Reláció KULCS

A Reláció KULCSA az ATTRIBÚTOM -ok Részhalmaza - tehát egy vagy több Attribútum.

Szerepük, a relációs adattáblák EGYED ELŐFORDULÁS -ainak egyedi azonosítása.

Amennyiben egyetlen attribútum nem azonosítja az egyed előfordulást, akkor több attribútum alkothatja az egyedi azonosítót. A KULCS ezek alapján tehát lehet, **EGYSZERŰ KULCS** - amikor csak
1 db, illetve **ÖSSZETETT KULCS** ha 2 vagy több Attribútum vesz részt az egyedi azonosító képzésben. Az Összetett kulcsból elhagyva bármelyik összetevőjét, a kulcs elveszti azonosító tulajdonságát.

Az egyszerű kulcsot nevezi a szakirodalom még MINIMÁLIS KULCS –nak illetve az összetett kulcsot SZUPER KULCS –nak is.

A Kulcsként funkcionáló Attribútumokat **ELSŐDLEGES ATTRIBÚTUM** -nak, a nem kulcs (leíró) Attribútumokat pedig **MÁSODLAGOS ATTRIBÚTUM** -nak is nevezzük.

A kulcsok jelölése az Attribútum felsorolásban, az aláhúzás folytonos vonallal

.
pl. - a TANULÓK { **TKÓD**, OKOD, NÉV, ÉLETKOR, LAKCÍM, stb.}

A kapcsos zárójelek között felsorolt Attribútumok közül a TKÓD a TANULÓK táblázat egyedi azonosítója s így KULCSA.

- a RENTETEL { **RSZAM, CIKKOD**, MENNYI, ÖSSZÉRT, stb. }

Attribútumok közül az RSZAM, CIKKOD a RENTETEL táblázat KULCSA.

A KÜLSŐ KULCS fogalma

A relációs adatbázis táblák közötti logikai kapcsolatot a KÜLSŐ KULCS –ként funkcionáló Attribútumok biztosítják.

A **KÜLSŐ KULCS** a táblázat azon Másodlagos (leíró) Attribútuma(i), amely(ek) egy másik táblázatban Kulcs, azaz egyedi azonosítója.

A Külső kulcs jelölése az Attribútum név aláhúzása szaggatott vonallal.

pl. a CIKKLISTA és a MELEIRÁS kapcsolatában:

CIKKLISTA { CIKKSZÁM, NÉV, ME, EGYSAR }
MELEIRÁS { **ME**, MENÉV)

a DOLGOZÓ tábla és a PRÉMIUM tábla kapcsolatában:

DOLGOZÓ { **DOLGKÓD**, NÉV, FIZETÉS }
PRÉMIUM { DÁTUM, DOLGKÓD, PRÖSSZEG }

A REKURZIV KÜLSŐ KULCS fogalma

A Relációs adatbázis modellben a külső kulcs engedélyezett ugyanabban a Táblázatban. Ezt nevezzük **REKURZIV KÜLSŐ KULCS –**nak.

pl. DOLGOZÓK tábla, amikor a felettes kódja is egy adat:

DOLGOZÓK { DKOD, DNÉV, BEOSZTÁS, FŐNÖK }

A külső kulcsoknak több változata is létezik.
Párhuzamos kapcs (GK és tulajdonos ill. üzembentartó)
1 : 1 kapcsolatban Személyi szám – Vezetői engedély szám
Egy tábla több táblához kapcsolása (cikkszám – vevőkód, szállítókód)

4.3. Funkcionális függőségek

Ez azt jelenti, hogy az Attribútumok egy csoportja, függ az Attribútumok másik csoportjától.

A függőség a Táblázat belső szerkezetét, az Attribútumok kapcsolatait írja le.

pl. a Személyi számtól egyértelműen függ a név és más személyes adatok, a Rendszámtól, a épkocsi és tulajdonosának adatai, a TKÓD Attribútumtól függ a névsor tábla többi adata,

Függőség fajták:

TELJES Függőség -ről akkor beszélünk, ha több Attribútum együttesétől úgy függ a többi Attribútum, hogy nincs köztük olyan, amelyiktől külön is függenének.

Ilyen például a {HELY, IDŐPONT, HŐMÉRSÉKLET, LÉGNYOMÁS } adatokat tároló táblában a {HELY, IDŐPONT } Attribútum csoport is.

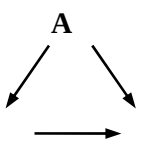
Ilyen a RENTETEL { RSZAM, CIKKOD } Attribútum együttes, mert a többi adat e kettő együttesétől függ és nem függ külön egyiktől sem.

De **NEM TELJES Függőség** a következő táblában,

TABLA { RSZAM, CIKKOD, CIKKNÉV, MENNY, ME, EGYSAR, stb.}
mert a (RSZAM, CIKKOD) -tól csak a MENNY függ, a többi adat csak a CIKKOD -tól függ.

TRANZITÍV Függőség: ha létezik olyan Attribútum, amely közvetlenül és láncoltan is függ egyetlen Attribútumtól

Van egy A, B, C attribútum együttes. A -tól függ a B és C, de függőség van még B és C között is.



B C

pl. a Dolgozók bér nyilvántartásában szerepelnek: { DKOD, DNÉV, HAVIBÉR, ÉVESBÉR }

a DKOD -tól függ minden egyéb adat, de az ÉVESBÉR még a HAVIBÉR -től is függ, tehát Tranzitív függés van a HAVIBÉR és ÉVESBÉR között.

Ilyen még például a { NÉV, IRSZ, VÁROS, CÍM }
NÉV → IRSZ, VÁROS ugyanakkor IRSZ → VÁROS

4.4. Redundancia

Redundáns adat az, amelyet többször is tárolunk az AB -ban (több táblában !), vagy számítható a tárolt adatokból.
Plusz helyet foglal, tehát kerülendő a redundancia. Ezen kívül még adat karbantartási problémákat is okozhat illetve az adatbiztonság is romlik (több helyen kell hozzányúlni ugyanahhoz az adathoz, esetleg elfelejtjük).

Ennek ellenére, a fizikai tervezés során, elsősorban a gyorsabb adatelérés céljából - tudatosan és meggondoltan - alkalmazhatjuk.

pl. { TANULÓ, HIÁNYZÁSA, HIÁNYOZHAT }

A HIÁNYOZHAT adatot felesleges minden tanulóhoz hozzárendelni, mivel mindegyiknél ugyanaz. Ezt tehetjük az ISKOLA táblázatba mert itt írjuk le az iskola jellemzőit.

Adatbázis tervezés – NORMALIZÁLÁS

Az AB optimális szerkezetének kialakítását NORMALIZÁLÁS -nak nevezzük.

A Normalizálás folyamata több (elméletileg 5) lépésből áll. Lépésenként mindig javítjuk az AB szerkezetét azaz 1. Normál Formára, 2. NF -ra, ... stb. hozzuk.

A gyakorlat megelégszik a 3. NF -ra hozott AB szerkezettel.

Figyelem ! Alapszabály, hogy azonos tartalmú egyed előfordulások (rekordok, sorok) TILTOTTAK a táblázatban !

1. NF -ra hozás:

A relációs AB csakis olyan Egyed előfordulásokat tud feldolgozni, amelyekben csak egy Attribútum érték van mezőnként.

Az 1. NF -ra hozott táblázatot elvileg már feldolgozhatjuk, mert az ABKR elfogadja.

pl. A DOLGOZÓ lista tartalmazza a
{ DKOD, DNEV, SZDAT és KÉPZETTSÉG } Attribútumokat

Egy dolgozónak azonban lehet több szakképzettsége is !

DKOD	DNEV	SZDAT	KÉPZETTSÉG
01	Kiss Ede	47.01.04	mérnök, közgazdász
02	Soós Márton	72.11.25	könyvelő
03	Nagy Béla	66.06.12	programozó

SZABÁLY: **az AB 1. NF -ban van, ha a táblázat minden attribútum előfordulás (egy mező) csak EGY adatot tartalmazhat.**

1. NF létrehozása: - két módszer lehetséges

A. módszer: növeljük a sorok számát

DKOD	DNEV	SZDAT	KÉPZETTSÉG
01	Kiss Ede	47.01.04	mérnök
01	Kiss Ede	47.01.04	közgazdász
02	Soós Márton	72.11.25	könyvelő
03	Nagy Béla	66.06.12	programozó

Az 1.NF -nak megfelel, viszont a tábla kulcsát (egyedi azonosítását) ki kell bővíteni:
{ DKOD, KÉPZETTSÉG } -re.

B. módszer: "szétvágjuk" a táblát két táblára

DKOD	DNEV	SZDAT
01	Kiss Ede	47.01.04
02	Soós Márton	72.11.25
03	Nagy Béla	66.06.12

DKOD	KÉPZETTSÉG
01	mérnök
01	közgazdász
02	könyvelő
03	programozó

Kiemeltük a DKOD azonosítót és az új táblában csak ezt és a kérdéses Attribútumot tároljuk.
Az első táblában megmaradt kulcsnak a DKOD.
A második táblában természetesen mindkét Attribútum kulcs.

További példa, a Megrendelés nyilvántartás, ahol az űrlap formátumú táblázatból A. eljárással a MEGREND táblázatot, a B. eljárással a RENDADAT és TETELEK nevű táblázatokra szétvágva hoztuk létre az 1. NF -t.
Az RSZAM -nak be kell kerülnie a TETELEK táblába is, mert csak ezen keresztül lehet biztosítani a két tábla kapcsolatát.

Lásd a mellékletet. (1. 2. 3. -as ábrák)

2. NF -ra alakítás:

A 2.NF –ra hozás megszünteti a MÓDOSÍTÁSI, TÖRLÉSI és BŐVÍTÉSI ANOMÁLIÁKAT !

SZABÁLY: az AB 2. NF -ban van, ha az már 1. NF -ban van és a táblázat másodlagos attribútumai az (összetett) kulcs egészétől függenek.
(vagyis egy több Attribútumból álló kulcs mindegyik elemétől függ, nem csak egyiktől vagy másiktól)

Következtetések:

- ha csak egy elemű a kulcsa egy 1.NF -ban levő táblázatnak, akkor az egyúttal már 2. NF -ban is van.

- ha a táblázatban nincsenek másodlagos Attribútumok, akkor az eleve 2. NF -ban van.

pl. a { DKOD, DNEVE, SZDAT, BEOSZTAS, stb.) táblázat, amelynek a DKOD a kulcsa (elsődleges Attribútum) a többi adat mind másodlagos Attribútum, természetesen 2NF -ban van.

Ha van egy olyan táblázatunk, amelyben a havi bérek és prémium összegeit tartják nyilván a következő adattartalommal,

{ ÉVEK, KATEGÓRIA, HAVIFIZ, PRÉMIUM),

az összegek nagysága, a HAVIFIZ és a PRÉMIUM is a ledolgozott évektől és a kategóriától együttesen függ,
akkor a tábla 2.NF -ban van .

DE !

ha a HAVIFIZ csak az ÉVEK -től függ, a PRÉMIUM pedig továbbra is mindkettőtől, máris 2.NF -ra kell hozni, azaz szét kell vágni

Figyelem ! Itt nem tranzitív függésről van szó, mert a kulcs összetett !

HAVI { ÉVEK, HAVIFIZ } és
PRÉMIUM { ÉVEK, KATEGÓRIA, HAVIFIZ } táblázatokra.

A 2. NF -ra hozás tehát az 1. NF -nak megfelelő táblázatok szétvágásával történik.

Folytatva a Megrendelések AB normalizálását a RENDADAT és TETELEK táblákat külön megvizsgáljuk, hogy szükséges -e tovább normalizálni. (4. ábra)

A RENDADAT tábla KULCSA : (RSZAM)

Ebből az következik, hogy a RENDADAT 2.NF -ban van.

De azért érezzük, hogy még nincs egészen rendben
? : muszáj itt tárolni ezeket az adatokat ?
R : valószínűleg redundancia, mert az végösszeg számítható

A TETELEK tábla KULCSA : (RSZAM, CIKKSZAM)

Megvizsgáljuk, hogy a sor adatai (az Attribútumok) csak ettől, az összetett kulcstól függenek -e, vagy van olyan Attribútum, amely valamelyik elemétől is függ közvetlenül.

Az (RSZAM, CIKKSZAM) -tól függ közvetlenül : a MENNYI és a Cikk össz

Csak a CIKKSZAM -tól függ : a CIKKNEVE, MEKOD, MENEV, EGYSAR és AFA

Tehát nincs 2.NF -ban, ezért bontsuk szét, 2 táblázatra !

Egyik tábla: neve legyen RENTETEL, ebben legyenek az
{ RSZAM, CIKKSZAM, MENNYI és Cikk össz } adatok,

Itt az (RSZAM, CIKKSZAM) együtt a kulcs, és a többi adat e kettőtől függ, tehát 2.NF -ban van.

Másik tábla: neve legyen CIKKEK, ebben legyenek a
{ CIKKSZAM, CIKKNEVE, EGYSAR, AFA, MEKOD, MENEV }

Itt csak a CIKKSZAM a kulcs, tehát eleve 2.NF -ban van.

A táblázatok teljesen megfelelnek a 2.NF követelményeinek és használhatóak is egy AB -ban, de "érezzük", hogy vannak olyan adatcsoportok, amelyek külön táblázatot igényelnek és vannak még redundáns adatok is. Ezért tovább kell normalizálni.

3. NF -ra alakítás:

SZABÁLY: az AB 3. NF -ban van, ha az már 2. NF -ban van és a másodlagos attribútumok között nincs tranzitív függés

A tranzitív függés lényegét lásd a korábban. **

Ha megvizsgáljuk a Megrendelések AB három, 2. NF -jú táblázatát, kettőben fellelhető tranzitív függés. (4. ábra)

RENDADAT táblázat: ebben mindjárt kettő is van,

- az SZNEV, IRSZ, VAROS, CIM Attribútumok, egyrészt az RSZAM –tól (mert ez a táblázat kulcsa), ugyanakkor közvetlenül az SZKOD -tól is függenek (mert egy SZKOD -hoz tartozó adatokról van szó).

- a KNEV, KBEOSZT, NEVNAP Attribútumok, szintén az RSZAM -tól (mert ez a táblázat kulcsa), ugyanakkor közvetlenül a KKOD -tól is függenek (mert egy KKOD -hoz tartozó adatokról van szó).

CIKKADAT táblázat: ebben csak egy tranzitivitás van.

- a MENEV Attribútum, a CIKKSZAM kulcstól (mivel ez a táblázat kulcsa), ugyanakkor közvetlenül a MEKOD -tól is függ (mert egy MEKOD -hoz tartozó adatról van szó).

RENTETEL táblázatban nincs tranzitív függőség.

A 3. NF -ra hozás, további táblázat darabolással történik.

RENDADAT táblázat 3. NF -ra hozás:

A tábla tartalma:

RSZAM, RENDATUM, TELJESIT, SZKOD, SZNEV, IRSZ, VAROS, CIM, KKOD, *KNEV, KBEOSZT, NEVNEP*, BIZTOS, Szla.össz

Az aláhúzott adatok, az SZKOD (mint egyedi azonosító) -hoz tartoznak, kiemeljük egy önálló táblázatba, amelyben a Szállítók adatait tároljuk, neve: SZALLIT .

A *dőlt betűs* adatok, a KKOD (mint egyedi azonosító) -hoz tartoznak, kiemeljük egy önálló táblázatba, amelyben a Kapcsolattartó személyek adatait tároljuk, neve: KAPCS .

A maradék táblázat most már kevesebb adatot tartalmaz.
Benne azonban megmaradtak az új táblákhoz kapcsoló KÜLSŐ KULCSOK (SZKOD és KKOD).
Az új táblázat neve: RENDELES

CIKKADAT táblázat 3.NF -ra hozás:

Tartalma CIKKSZAM, CIKKNEVE, MENNYI, MEKOD, MENEV, EGYSAR, AFA,

A CIKKADAT -ban levő tranzitív függés, a MEKOD → MENEV adat párnál van.

A CIKKADAT -ból tehát kiemelünk egy MELEIRAS nevű , MEKOD, MENEV tatalmú táblát.

A táblázatok most már 3. NF-ban vannak de még meg kell vizsgálni a redundanciákat.

REDUNDANCIA vizsgálat: (5. ábra)

Adat többszörözés :nincs az táblázatokban (kétszer ugyanaz a másodlagos Attribútum nem szerepel bennük). A kulcsok ismétlődése nem redundancia, mert a tábla kapcsolatokat biztosítják.

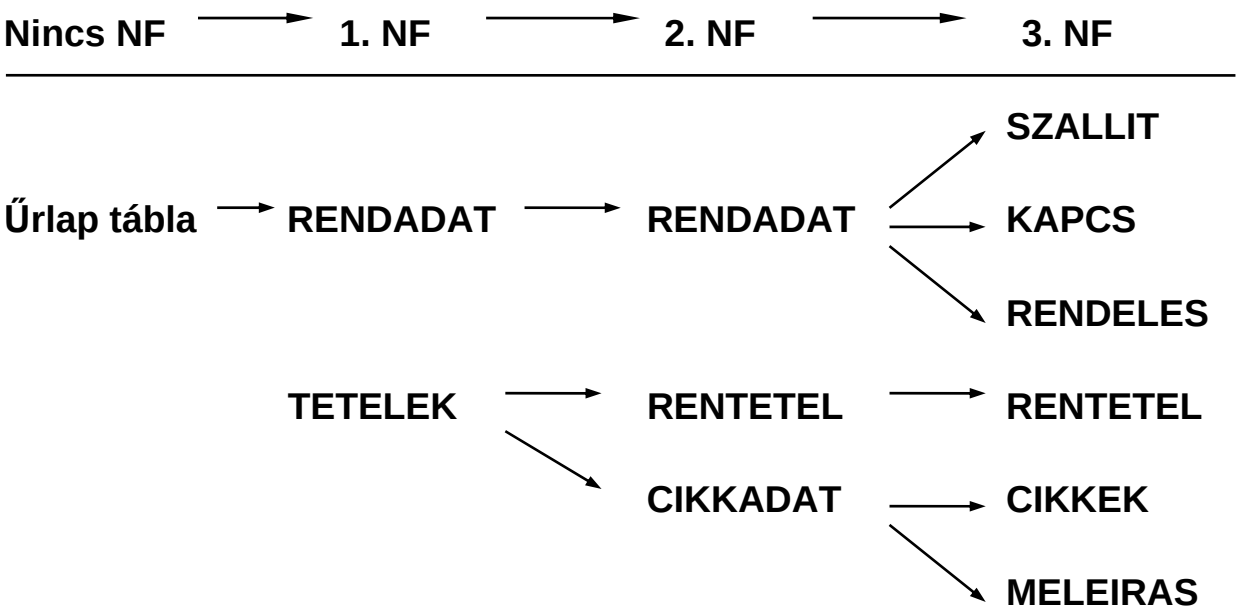
Számított adatok, azonban vannak.

- a RENTETEL táblában a Cikk össz , a MENNYI * EGYSAR * (1 + AFA) képlettel számítható, tehát felesleges tárolni

- a RENDELES táblában a Szla.össz szintén számítható (az RSZAM kapcsolaton keresztül) a RENTETEL rekordok Cikk össz adatainak halmozásával.

Dönteni kell tehát, a számított adatok elhelyezéséről a táblázatokban.
(Itt, most nem szerepeltetjük őket.)

A táblák normalizálási folyamata:



Fizikai AB tervezés: (lásd dBASE és ACCESS gyakorlat)

A normalizálást követi az AB fizikai tervezése, az adatszerkezet (Típus, Hossz) meghatározása, a táblázatok kapcsolatának megtervezése, az indexek (UNIQUE és általános index) maghatározása.

Mindezeket szigorú dokumentálási szabályok figyelembevételével tesszük.
Rajzos, táblázatos és szöveges leírás rögzíti az AB végleges szerkezetét.

A további feladatok már az AB használatával kapcsolatosak, (mit, mikor és hogyan kell karbantartani, milyen VIEW -k szükségesek a lekérdezésekhez, milyen adatvédelmi előírásokat kell betartani, stb.)

Adatvédelmi módszerek az ABKR –ben

A több felhasználós, hálózati üzemeltetési körülmények és az adatbázis tartalmi értéke különösen fontossá teszi az adatok védelmét sérülés és jogosulatlan hozzáférés szempontjából.

Az AB –t veszélyeztető tényezők:

1. Fizikai meghibásodás: ezek hardver oldalról következhetnek be, adathordozó, átviteli vonal, feldolgozó gép meghibásodásából erednek.

   Kivédésük: szünetmentes áramforrás,
   az átviteli vonalak kábelének védelme,
   adattárak és/vagy adatállományok tükrözése,
   biztonsági másolatok készítése,
   rendszeres, tervszerű karbantartás

2. Adatátviteli hibák: hardver (hálózatszakadás) vagy szoftver (op. rendszer) hibák miatt az I / O műveletek során sérül az adat.

   Kivédésük: hálózati csomópontok többirányú elérhetősége,
   a HW és az Operáció és Hálózati Rendszer összehangolása,
   ellenőrző összegek alkalmazása,

3. Jogtalan hozzáférés: az AB –t megnyitni vagy az AB elemeihez csak a megfelelő jogosultság alapján lehet hozzáférni

A hozzáférési jog: az AB megnyitásának jogosultsága

A hozzáférési jogokat a Rendszergazda (adatbázis adminisztrátor) jogosult kiadni. A jogosultság a bejelentkező (az AB –t megnyitó) felhasználói nevéhez és jelszavához van csatolva. A jogosultság lehet csoportos is (workgroup).
Az operációs rendszerbe bejelentkezőhöz hozzárendelhető egy DEFAULT adatbázis, amelybe nem kell külön bejelentkeznie a felhasználónak, rögtön használhatja azt.

Megj.: a továbbiakban az **SQL92 szabvány** szerinti utasításkészlet kerül bemutatásra, amely egy-egy konkrét ABKR esetén kis mértékben el is térhet.

CONNECT TO nyitandó AB neve USER felhasználónév, jelszó
DISCONNECT { AB név | ALL | CURRENT } - {az AB nevűt | összes nyitottat | éppen aktívat}
SET CONNECT TO { DEFAULT | a másik nyitott AB neve } - átlépés másik, nyitott AB -ba

Felhasználói jogok: a megnyitásra jogosult milyen állományokkal, milyen műveleteket végezhet el.

A felhasználói jogosultság kiterjed az állományokhoz való hozzáférés (mely állományokat láthat valaki a megnyitott AB -ból) illetve az állományokon végezhető műveletek (csak olvashat, módosíthat is, törölhet) korlátozására.
Ezen jogok is a rendszergazda által kerülnek rögzítésre a megnyitási joggal együtt. Ezek bármikor módosíthatók is.

A jogok kiadása:

GRANT műveleti jog ON objektum TO { PUBLIC | felhasználó1, …. felhasználóN }
[WITH GRANT OPTION]

műveleti jog: a kijelölt objektumon milyen művelet (olvasás, írás, stb.) elvégzésére van jogosultsága

objektum: alapértelmezése mindig TABLE
de lehet CHARACTER SET - a használt kódtábla megadása
COLLATION - a lokális rendezési sorrend definiálása
DOMAIN - közös tulajdonságtípus jellemzők változtatása
TRANSLATION - két kódtábla megfeleltetése tárolt eljárások módosítási engedélye

PUBLIC : minden bejelentkezési joggal rendelkezőre vonatkozó hozzáférési jog

WITH GRANT OPTION : a jog továbbadása a bejelentkező részére

A jogok visszavétele:

REVOKE műveleti jog ON objektum FROM { PUBLIC | felhasználó1, …. felhasználóN }

4. Adatértékekre vonatkozó szabályok megsértése: az adatforgalom során eltérő de megengedett adattípus csere fordulhat elő, illetve adatstruktúra módosításkor adatvesztés léphet fel.

Kivédése: az azonos adat típusú Attribútumok (oszlopok) csoportba foglalása – DOMAIN
körültekintő struktúra módosítás

5. Inkonzisztencia: az összefüggő állomány módosítások részleges végrehajtása (egy tranzakcióval több állományt is módosítani kellene) illetve logikai állomány összefüggések figyelmen kívül hagyása (az egyik állományba nem áll fenn bizonyos feltétel, ami a
másik állomány módosításához szükséges)
- másként: állomány kapcsolati szabályok megsértése

Kivédése: körültekintő jól letesztelt programmal
az SQL speciális naplózó lehetőségével, ami arra jó, hogy hiba esetén tetszőleges lépésig vissza lehet állítani a kiinduló állapotot
(a 4. pontnál is alkalmazható)

Irodalom: SQL Kézikönyv - ComputerBooks, Budapest, 1998. 151. oldal