ADATBÁZIS-KEZELÉS AB-1

Az adatbázis-kezelés alapjai

Adat és információ

Adatbázisok

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

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

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

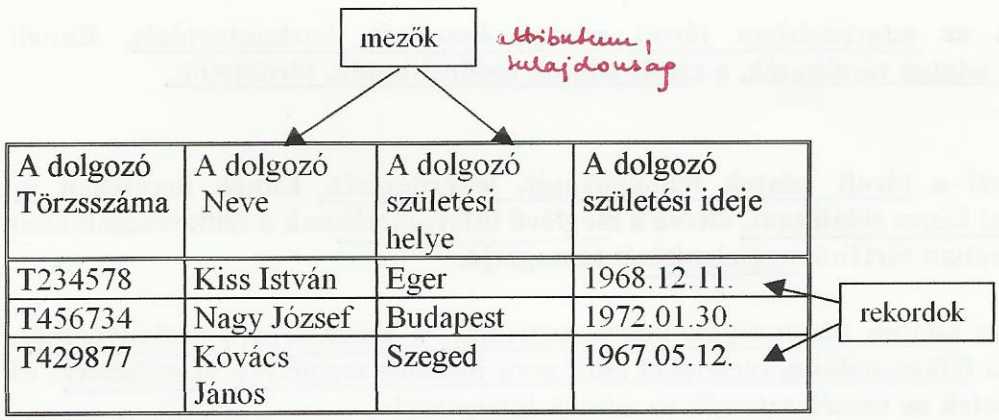
Adatbázis adminisztrátoroknak.

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

Metaadatok

1. Példa

Egy vállalat dolgozóinak adatait nyilvántartó program adatbázisának része lehet.

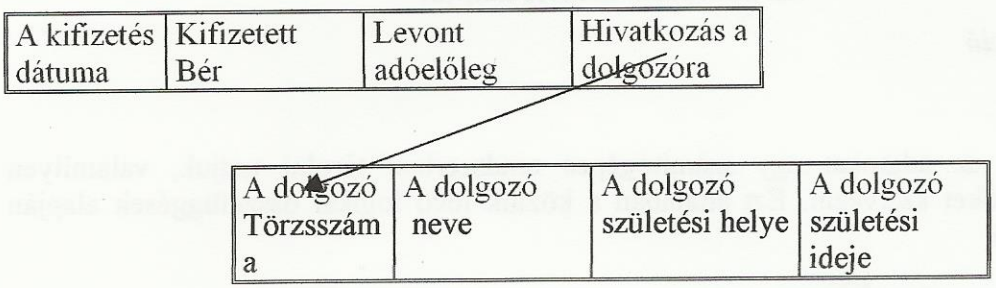


A dolgozó Törzsszáma	A dolgozó Neve	A dolgozó születési helye	A dolgozó születési ideje
T234578	Kiss István	Eger	1968.12.11.
T456734	Nagy József	Budapest	1972.01.30.
T429877	Kovács János	Szeged	1967.05.12.

Adatredundancia

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

Példa



2. *Adatbázis-kezelő rendszerek*

Az AB kezelésére speciálisan erre a célra kifejlesztett programok léteznek:

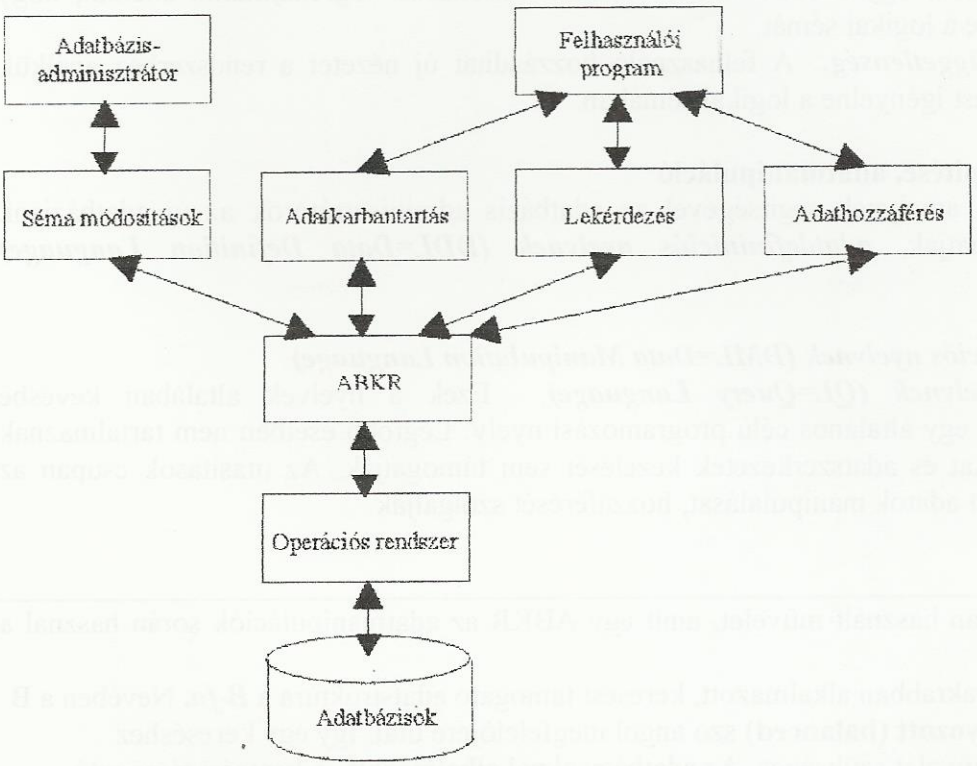
A*datbázis-kezelő rendszernek* (ABKR).

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

Egy ABKR általában a következő feladatokat látja el:

- **Támogatja új adatbázisok létrehozását, azok struktúrájának, tárolási módjának kialakítását.**
- **Megvalósítja az adatbázisban tárolt adatok kezelését, karbantartását. Ennek keretében új adatok tárolhatók, a tárolt adatok módosíthatók, törölhetők.**
- **Lehetővé teszi a tárolt adatok feldolgozását, lekérdezését. Ennek keretében új információkat képes előállítani, illetve a meglévő információknak a felhasználók által igényelt formában történő megjelenítését támogatja.**
- **Garantálja az adatok biztonságát, konzisztenciáját, a hozzáférések szabályozását, vagyis hogy a felhasználói műveleteket csak arra jogosult személyek végezhessék, és ezek a műveletek ne veszélyeztessék az adatok integritását.**
- **Lehetővé teszi az adatbázisok megosztását több felhasználó között.**

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



Az ANSI/SPARC modell

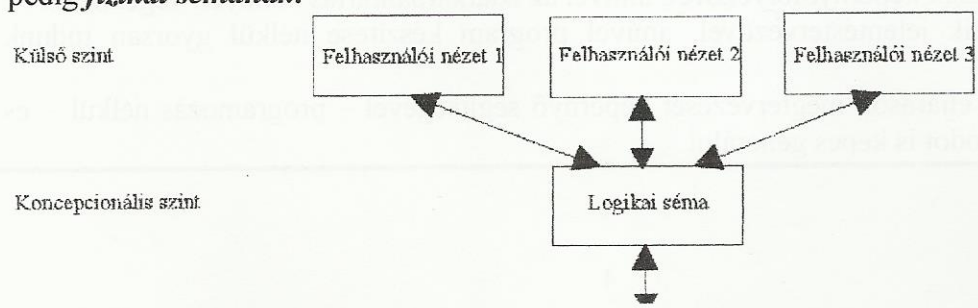
Láthatjuk az előző ábrán, hogy az ABKR egyfajta összeköttetést, hidat jelent a felhasználó és a számítógépes háttértárolón fizikailag tárolt adatok között. Az ***Amerikai Szabványügyi Hivatal (ANSI=American National Standards Institute) Szabvány Tervezési és Követelmények Bizottsága (SPARC=Standards Planning And Requirements Committe)*** ezt az összeköttetést három szintre osztotta fel.

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

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

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

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



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

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

Adatbázisok felépítése, adatmanipuláció

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

Adatmanipulációs nyelvnek (DML=Data Manipulation Language).

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

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

Az egyik leggyakrabban alkalmazott, keresést támogató adatstruktúra a ***B-fa.*** Nevében a B betű a **kiegyensúlyozott (balanced)** szó angol megfelelőjére utal. Így egy kereséshez minimális lemezművelet szükséges. Az adatbázisoknál alkalmazott, a keresést támogató adatstruktúrát ***indexnek*** nevezzük. Az adatbázis sémájánál meg kell adni, hogy mely mezők alapján kívánunk indexeket készíteni. Az index arra szolgál, hogy a keresési és rendezési műveleteket gyorsabbá tegye. De a beszúrás, törlés, módosítás műveleteknél inkább lassító hatása van, mivel ezeknél az index adatstruktúrát is aktualizálni kell. Használata olyan adatbázisoknál előnyös, amelyeknél kevés módosítás történik, viszont gyakran kell keresni.

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

Képernyőtervező (screen generator).

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

Jelentéstervezőnek (report generator)

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

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

A 4GL rendelkezik képernyőtervezővel, amivel az adatkarbantartás könnyen végezhető.
A 4GL rendelkezik jelentéstervezővel, amivel program készítése nélkül gyorsan tudunk listákat készíteni.
A 4GL támogatja eljárások megtervezését képernyő segítségével – programozás nélkül – és ezekből programkódot is képes generálni.

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

Adatintegritás

Az adatok korrektek, konzisztensek és aktuálisak.

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

Az adatbázis a tényleges adatok mellett tárolja azok összefüggéseit, kapcsolatait is. ***Hivatkozási integritás.*** Ez azt jelenti, hogy egy kapcsolatnál a hivatkozott adatnak létezni kell. A korszerű ABKR az adatmanipulációk során ellenőrzi, hogy a hivatkozási integritást nem sérti-e a művelet.

Az adatok helyessége nemcsak az ABKR felelőssége, az alkalmazás fejlesztőjének is igyekeznie kell úgy megterveznie az adatbázist, hogy abban a lehető legkevesebb lehetőség legyen arra, hogy az adatok valamilyen szempontból helytelenek legyenek. Például tegyük fel, hogy egy alkalmazás adatbázisában magyarországi városok szerepelnek. Megtervezhetjük úgy az adatkarbantartást, hogy a felhasználónak kell mindig begépelnie a megfelelő város nevét. Ez azonban sok hibalehetőséget rejt magában. Az adatot felvivő személy hibát véthet a gépelés során, illetve előfordulhat – különösen hosszabb nevű városok esetén – hogy egyik alkalommal rövidített, más alkalommal teljes formában gépeli be az adatot. A számítógép számára azonban a különbözőképpen írt nevek különböző várost jelölnek. Ez problémákat okozhat abban az esetben, ha például városok szerint kell összesíteni, vagy rendezni adatokat. Megteheti azonban az alkalmazás fejlesztője, hogy már a **fejlesztés során összegyűjti a lehetséges városok neveit, s azokat eltárolja az adatbázisban.** Adatbevitelkor pedig csak választani kell közülük. Ezáltal a hibázás lehetősége jelentősen csökken.

Adatvédelem

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

Egy jó ABKR-nek gondoskodnia kell arról, hogy lehetőség legyen az adatok védelmére. Ha ezt nem lehetne megvalósítani, nagy bizalmatlanságot keltene a felhasználóban, és jelentősen korlátozná az adatbázis-alkalmazások elterjedését.

A második problémát az adatbázisban bekövetkező esetleges sérülések jelentik. Ez több okból történhet. Előfordulhat hardware meghibásodás, de a szoftverekben is bekövetkezhet olyan hiba, amely az adatok károsodását eredményezheti. Sőt akár egy-egy rendkívüli esemény, mondjuk egy áramszünet is okozhat ilyen problémát.

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

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

Adatbázisok megosztása

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

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

3. Példa

Tegyük fel, hogy egy bérszámfejtő rendszerben ketten dolgoznak. Az egyik felhasználó a fizetésemeléseket rögzíti, a másik pedig éppen egy olyan alkalmazást futtat, ami újraszámolja az adókat. Feltételezzük, hogy a második program kiolvassa a régi fizetést egy adott személy esetén az adatbázisból, s miközben az adószámítási eljárás fut, az első program módosítja a fizetést és az adót is. A második eljárás azonban erről mit sem tud, tehát amikor kész van, visszaírja az adatbázisba az adót. Ez azonban még a régi fizetés adója, ami kevesebb, mint a tényleges adó.

Kizárólagos joggal való megnyitás.

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

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

4. Adatmodellezés

Adatbázis tervezés

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

A tervezés maga egy kreatív folyamat. Mivel az alkalmazások nagyon különbözőek lehetnek, nincs olyan általános tervezési technika, ami mindig egy az egyben alkalmazható lenne. Egy jó terv elkészítéséhez megfelelő tapasztalat szükséges, amelyet gyakorlattal lehet megszerezni. Táblázatos formában foglaljuk össze egy alkalmazás tervezésének legfontosabb fázisait.

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

Az információ igény meghatározásának fázisában az egyik fő feladat az alkalmazás céljainak meghatározása. A célok szoros összefüggésben vannak a felhasználói igényekkel. Ezért a tervezésnek ebben a fázisában a felhasználókkal való szoros kapcsolattartás elengedhetetlen. Szükséges az is, hogy a tervező járatos legyen a rendszer felhasználóinak szakterületén. A tervezést általában nem a programozó végzi, hanem külön a tervezésben gyakorlott személyek:

szervezők.

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

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

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

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

Viszonylag jobban elkülönül az előző két fázistól a fizikai tervezés szakasza. Ez tulajdonképpen az alkalmazás megvalósítását jelenti a logikai tervek alapján. A gyakorlatban a szervező egy tervdokumentációban rögzíti a rendszertervet, amelyben leírja az adatbázis

pontos tervét, az adatok beviteli és megjelenítési formátumát, az algoritmusokat. Ennek alapján készíti el a programozó az adatbázis-rendszert

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

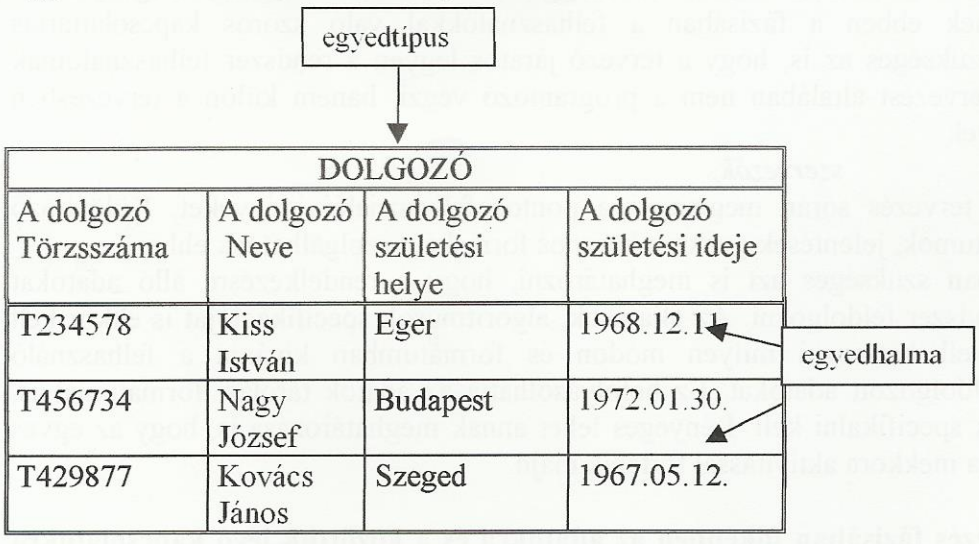
Adatmodellezés alapelemei.

Az egységes tervezés érdekében kidolgoztak modelleket, amelyeket ***adatmodelleknek*** szokás nevezni. Egy adatmodell alapvetően nem magukkal az adatokkal foglalkozik, hanem azok struktúrájával, a közöttük lévő összefüggésekkel, kapcsolatokkal. A legelterjedtebb adatmodellezési technikák közös jellemzője, hogy három alapelemből tevődnek össze. Ezek a következők:

- **Egyed, vagy egyedtípus**
- **Tulajdonság, vagy tulajdonságtípus**
- **Kapcsolat, vagy kapcsolattípus**

Egyed

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

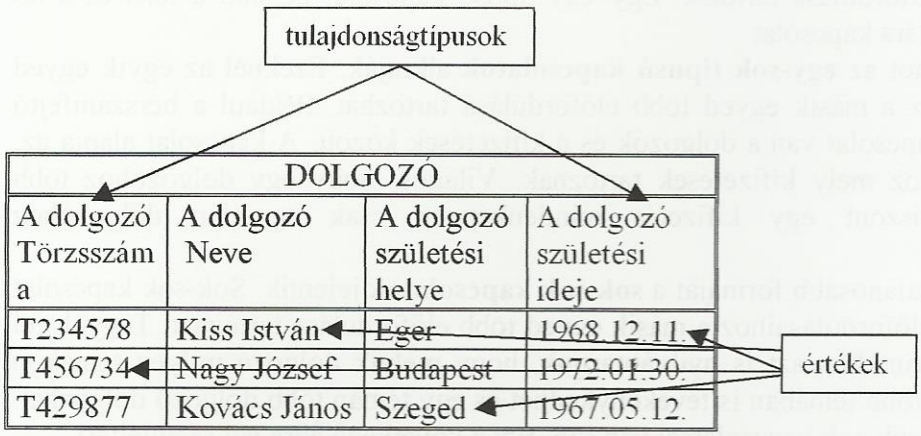


Az egyed előfordulások a rekordoknak felelnek meg. A gyakorlatban az egyedtípust szokás ***rekordtípusnak*** is nevezni. (rekord- vagy struktúratípus).

Tulajdonság

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

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



KULCS. Fontos szerepe van azoknak a tulajdonságoknak, amelynek értékei a többi tulajdonság értékeit egyértelműen meghatározzák. Ez azt jelenti, hogy ha az ilyen tulajdonságok értékeit megadjuk, akkor az egyértelműen definiál egy előfordulást. Azokat a tulajdonságokat, amelyek egyértelműen meghatározzák az egyedtípus egy elemét, ***kulcsnak*** nevezzük. A kulcsok fontos szerepet töltenek be az adatmodell kialakításánál. A tervezés során általában meg szokták adni, mely attribútumok fogják a kulcsokat alkotni. Elvileg egy egyednek több kulcsa is lehet, de a legtöbb esetben egyet szokás kiválasztani, amely a leginkább alkalmas az egyértelmű azonosításra. Ezt hívjuk ***elsődleges kulcsnak.***

Kulcsjellegű tulajdonság mindig található. Ha a tényleges adatok között nem lenne ilyen, akkor bevezethetünk egy olyan tulajdonságot, amelynek értékei sorszámok, kódszámok, speciális azonosítók. Ez betöltheti az elsődleges kulcs szerepét.

Láthatjuk hogy az azonosítók, kódszámok szinte minden alkalmazásnál előfordulnak. A számítógép jellegéből adódóan ezek alkalmasak az előfordulások pontos meghatározására. Bizonyos esetekben a laikus felhasználó számára nehézséget okozhat, hogy a kódnál már egy apró elírás is egész más eredményt szolgáltathat. Aki számítógéppel dolgozik, annak tudomásul kell venni a kódok használatát, és pontos munkára kell törekednie.

Kapcsolat

Az adatmodell harmadik fontos elemét a ***kapcsolatok*** jelentik. Kapcsolatnak nevezzük az egyedek közötti összefüggést, viszonyt. Például a már jól ismert bérszámfejtő rendszerben a dolgozó és a kifizetés egyedek között létezik egy természetes kapcsolat. Ez azt mondja meg, hogy az egyes dolgozókhoz mely kifizetések tartoznak.

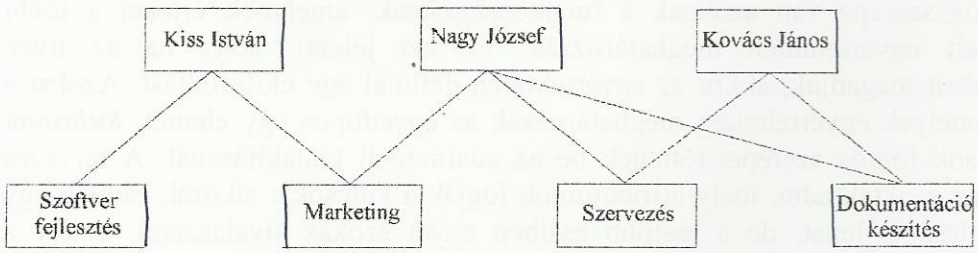
A fentiek alapján kapcsolatok az egyedhalmazok elemei között alakíthatók ki. **Osztályozhatjuk a kapcsolatokat aszerint, hogy egy-egy elemhez hány másik elem tartozik.** Ennek a kapcsolat számítógépes reprezentációja szempontjából van jelentősége. Sokkal egyszerűbb ugyanis egy olyan kapcsolatot megvalósítani, ahol egy egyed előforduláshoz csak egyetlen másik egyed előfordulás tartozhat, mint ha több. Az előbbinél elegendő lehet egy **mutató**, míg az utóbbinál valamilyen összetett adatstruktúrára van szükség, például **halmazra,** vagy **listára.** A kapcsolatokat ezek alapján három csoportba szokás sorolni:

- **Egy-egy típusú**
- **Egy-sok típusú**
- **Sok-sok típusú**

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

A következő csoportot az **egy-sok típusú kapcsolatok** alkotják. Ezeknél az egyik egyed minden előfordulásához a másik egyed több előfordulása tartozhat. Például a bérszámfejtő rendszerben egy-sok kapcsolat van a dolgozók és a kifizetések között. A kapcsolat alapja az, hogy melyik dolgozóhoz mely kifizetések tartoznak. Világos, hogy egy dolgozóhoz több kifizetés tartozhat, viszont egy kifizetés mindenképpen csak egyetlen dolgozóhoz kapcsolódik.

A kapcsolatok legáltalánosabb formáját a **sok-sok kapcsolatok** jelentik. Sok-sok kapcsolat esetén mindkét egyed előfordulásaihoz a másik egyed több előfordulása tartozhat. Tegyük fel hogy dolgozói rendszerünkben azt is nyilvántartjuk, hogy melyik dolgozó milyen témákon dolgozik. Egy dolgozó több témában is tevékenykedhet és egy témán több dolgozó dolgozhat. Ebben az esetben tehát sok-sok kapcsolatról van szó. Ezt a következő ábra ezt szemlélteti.

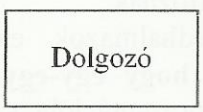


A sok-sok kapcsolatok láthatóan egy-sok kapcsolatokon alapszanak. Ha bármelyik egyed szempontjából nézzük, akkor egy-sok kapcsolatot fedezhetünk fel. **Ezért minden sok-sok kapcsolat felbontható két egy-sok kapcsolatra.**

Az eddigiekben olyan kapcsolatokról beszéltünk, amelyek két egyed között létesíthetők. Ezek az úgynevezett **bináris kapcsolatok.**

Egyed-kapcsolat diagrammok

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



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