ADATBÁZIS-KEZELÉS

elıadÆs vÆzlat

Összeállította: Várady Lajos
varadyl@math.klte.hu

Tartalom

Ajánlott irodalom

Quittner Pál: Adatbázis-kezelés a gyakorlatban, Akadémiai kiadó, Budapest, 1993

Dr. Halassy Béla: Az adatbázisok kezelésének alapvetõ kérdései, 1978, Budapest, KSH

Priskinné R. Zsuzsa és Erdélyiné: Építsünk könnyen és lassan adatmodellt, 1997, Veszprém

Váthy Ágnes, Németh Krisztián: Adatmodellezési feladatok I., Veszprém 1996, Veszprémi Egyetem

Stolnicki Gyula: SQL kézikönyv, ComputerBooks (http://www.computerbooks.hu/sql.htm)

Balogh Judit, Rutkovszky Edéné: SQL Példatár, 1994

Ecsedi-Tóth Péter, …: Az ORACLE relációs adatbázis-kezelõ rendszer, 1990, Budapest, IQSoft Rt.

Juhász I., Almási B., Márton Á., Balogh J.,: ORACLE 6.0 referencia kézikönyv

http://pc123a.math.klte.hu/varadyl/oktatas.htm

Bevezetés

A következõ tárgyalásra azért van szükség, hogy a kurzus végére magunk is el tudjunk készíteni egy jól mûködõ adatbázist a probléma felvetésétõl elindulva egészen a fizikai megvalósításig.

A jegyzet struktúrája	⇔	**Az adatbázis-tervezés fázisai**
1. Alapfogalmak		1. A feladat specifikációja
1. ER modell, egy adatbázis-kezelõtõl független magas szintû adatmodell megismerése	⇔	2. Adatbázis-kezelõtõl független magas szintû adatmodell elkészítése
1. A fõbb adatbázis-kezelõtõl függõ alacsonyabb szintû adatmodellek megismerése (hálós, hierarchikus, relációs)	⇔	3. Adatbázis-kezelõ rendszer kiválasztása
1. Az ER modell leképezési szabályai a hálós, hierarchikus, relációs adatmodellre.	⇔	4. A 2. fázisban elkészült modell leképezése a kiválasztott adatbázis-kezelõ rendszernek megfelelõ (hálós, hierarchikus, relációs) modellre
1. Fizikai tárolás és elérési mód	⇔	5. Fizikai tervezés
1. Az SQL nyelv	⇔	6. Megvalósítás

1. Miért van szükség adatbázis-kezelõkre

A hagyományos adatkezelésnek számos hátránya van:

- Többszörös adattárolás (redundancia) és az adatok ellentmondásossága, adat-összeférhetetlenség (inkonzisztencia).
- Rugalmas változtathatóság hiánya és magas karbantartási igény.
- Felhasználói logikából nem következõ feldolgozási vagy programozási többlet.
- Az adatvédelem kívánt szintje hagyományos eszközökkel nem biztosítható.

A fenti problémák csak az adatbázis szemlélettel vagy az adatbázis-kezelõ rendszerek alkalmazásával küszöbölhetõk ki.

2. Adatbázis szemlélet

Ez annak az elismerését jelenti, hogy az adatok erõforrásoknak tekinthetõk, ezeknek erõforrás jelleget tulajdonítunk.

Erõforrásokat a következõk jellemzik:

- biztosításuk idõt és pénzt igényel
- szûkös természetûek
- racionális felhasználásuk elõbbre visz.

Ilyen értelemben az adatbázis-kezelés nem más, mint az adatokkal, mint erõforrásokkal történõ gazdálkodás. Az adatbázis-kezelõ rendszer mint szoftver pedig ennek az erõforrás gazdálkodásnak egy bizonyos értelemben vett, automatizált eszköze.

3. Adatbázis rendszer

3.1 Adatbázis rendszer (ABR/DBS[1])

Az adatbázis rendszer részei a következõk:

- az adatbázis (db=database),
- az adatbázis-kezelõ rendszer (dbms=database management system),
- az adatbázis-adminisztrátor (dba=database administrator), és
- a felhasználói környezet együtt.

3.2 Adatbázis

- **Adatoknak és a köztük lévõ kapcsolatoknak a tárolt rendszere.**
- **Egy adatbázisba valamilyen szempontból rokon adatok tartoznak.**
- **Az adatbázis integrált:** több felhasználó és/vagy több felhasználás adatait tárolja együtt.
- **Az adatbázis osztott:** Az adatbázishoz több felhasználó férhet hozzá, akár azonos idõben.

[1] Data Base System

- **Fizikai felépítése** (adatbázisfájlok + adatszótár).

Adatszótár (CDD)[2]:

Az adatszótár információkat tartalmaz a:

- rendszer objektumairól ill. azok részeirõl
- az egyes objektumokhoz való hozzáférési jogokról
- az objektumok közötti függõségekrõl, kapcsolatokról

3.3 Adatbázis-kezelõ rendszer (DBMS[3])

3.3.1 Feladata

Olyan professzionális szintû szoftver, amely az adatbázissal kapcsolatban a következõket tudja tenni:

- adatbázis-kezelési feladatokat (az egész adatbázis létrehozása, adatok visszakeresése, tárolt adatokhoz új adatok hozzávétele, tárolt adatokból bizonyosak törlése, tárolt adatokból bizonyosak módosítása, rendezés, ûrlapgenerálás, jelentéskészítés stb.)
- adatok közti komplex kapcsolatok létrehozását
- többféle hozzáférési módot
- osztott adatbázisoknál a szinkronizációt
- adatok védelmét (illetéktelen felhasználótól)
- adatok integritását (a jogosult felhasználók se tudják elrontani az adatbázist)
- helyreállíthatóságot
- adatfüggetlenséget
- eszközfüggetlenséget.

3.3.2 Komponensei

DDL (Data Definition Language/adatdefiníciós nyelv)

DML (Data Manipulation Language/adatmanipulációs nyelv)

DCL (Data Control Language/adatvezérlõ nyelv)

QL (Query Language/Lekérdezõ nyelv)

Forms (Ûrlapgenerátor)

Report (Jelentéskészítõ)

3.4 DBA[4]/Adatbázis adminisztrátor

Feladatai:

- az adatbázis megszervezése: adatmodell kialakítása, az adatbázis objektumainak definiálása, keresési stratégiák megválasztása (indexelés), jogosultságok adása és szabályozása,

---

[2] Common Data Dictionary lásd Quittner Pál: Adatbázis-kezelés a gyakorlatban, Akadémiai kiadó, Budapest, 1993, 7.10 fejezet

[3] Data Base Management System

[4] Data Base Administrator

- adatbázis-kezelõ szoftverkomponenseinek kezelése
- adatbázis karbantartása, konzisztencia biztosítása

3.5 ABR architektúra három szintje

- Külsõ szint
- Koncepcionális szint
- Belsõ szint

3.5.1 Külsõ szint (séma)

Külsõ szint az, ahogy az egyes felhasználók látják az adatbázist. Ez többnyire a koncepcionális szinten leírt objektumok része.

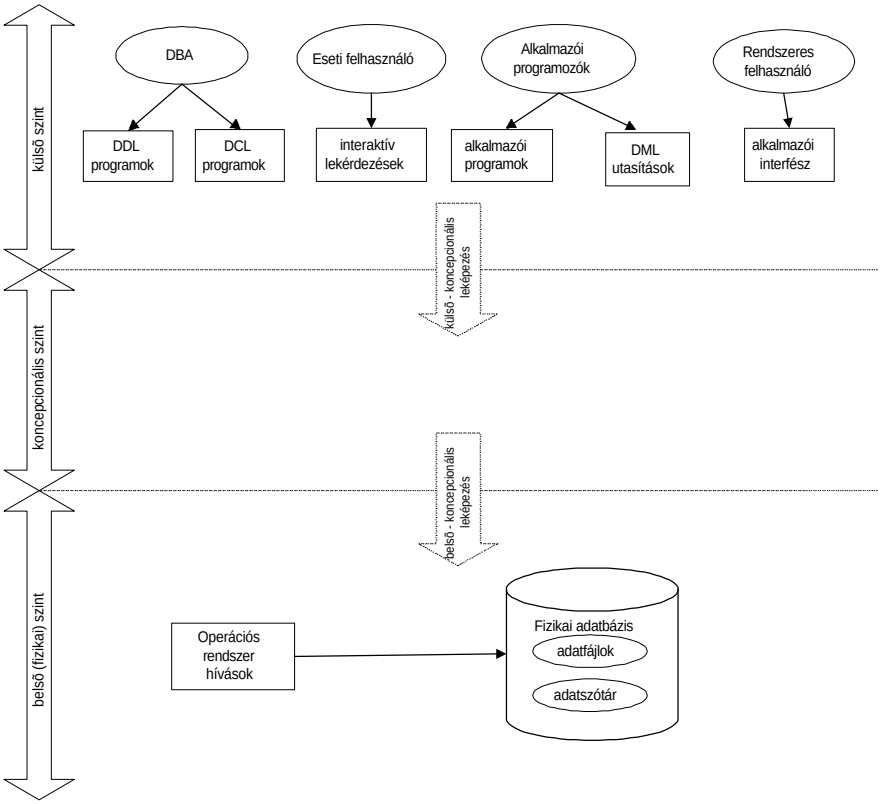
3.5.2 Koncepcionális szint (séma)

Koncepcionális szint írja le miként néznének ki mindenki számára az adatok. Ezen a szinten írjuk le az adatbázis összes objektumának szerkezetét, és a közöttük lévõ kapcsolatokat, valamint az objektumokhoz történõ hozzáféréseket.

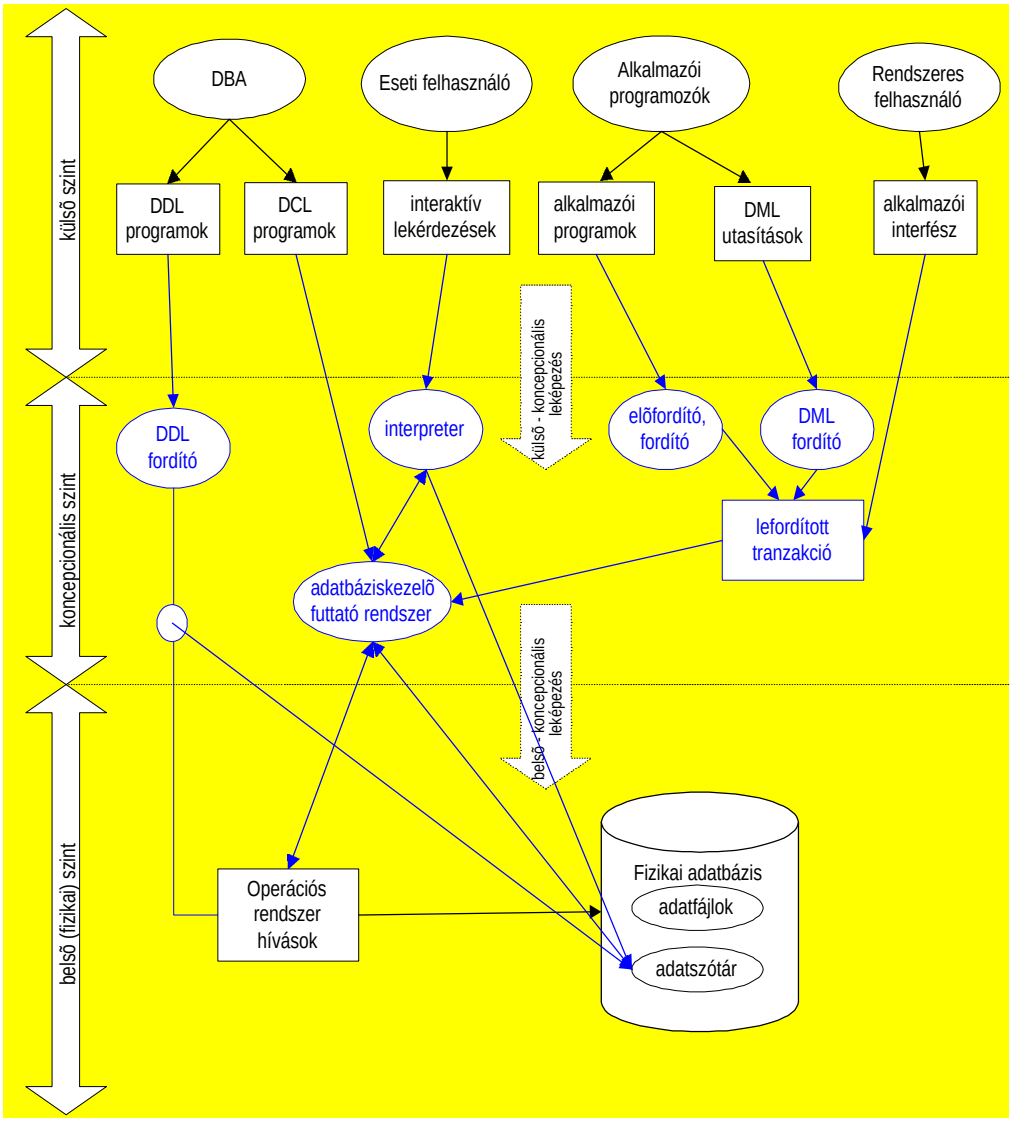
3.5.3 Belsõ szint

Leírja a fizikai tárolás és az elérés módját.

Ábra 1 Egy ABR architektúra 3 szintje (egyszeû)



Ábra 2 Ábra 3 Egy ABR architektúra 3 szintje (részletes)



3.6 Adatfüggetlenség

3.6.1 Logikai adatfüggetlenség

Az adatbázis logikai szerkezetében végrehajtott változások az adatbázist felhasználói programokat nem befolyásolják. A külsõ szint és a koncepcionális szint független egymástól.

3.6.2 Fizikai adatfüggetlenség (eszközfüggetlenség)

Az adatok tárolási és elérési módjának megváltozása nem vonja maga után az adatbázis logikai szerkezetének és a felhasználói programoknak a megváltoztatását. A belsõ szint független a koncepcionális és a külsõ szinttõl.

Teljes adafüggetlenségrõl/adatfüggetlenségrõl a logikai és fizikai adatfüggetlenség együttes megléte esetén beszélünk.

4. Adatmodellek

4.1 Adatmodell

Nem a konkrét adatokkal, azok elõfordulásaival, hanem azok típusaival illetve a közöttük lévõ kapcsolatokkal (egyedtípus, tulajdonságtípus, kapcsolattípus) foglalkozik, tulajdonképpen egyedek, tulajdonságok és kapcsolatok halmaza.

Egy adatbázis-kezelõ rendszer mindig egy adatmodellre épül.

Az adatmodellnek három szintje van:

- Külsõ
- Koncepcionális
- Belsõ

A külsõ és a koncepcionális szint adatmodellje alkotja a koncepcionális/logikai adatmodellt, a belsõ szint a fizikai adatmodellt.

Logikai mod.	Magas szintû (dbms-tõl független)	**ER** (Entity Relationship) modell
		EER (Enhanced Entity Relationship) modell
	Alacsony szintû (dbms-tõl függõ)	**Hierarchikus** modell
		Hálós modell
		Relációs modell

A magas szintû adatmodellbõl az alacsony szintû adatmodellre való áttérést ún. leképezési szabályok teszik automatikussá (CASE[5] TOOLS).

4.2 Az adatmodellek alapelemei

4.2.1 Egyedtípus (entitás)

Minden olyan objektum, ami minden más objektumtól megkülönböztethetõ, amirõl adatokat tárolunk, és amit tulajdonságaival kívánunk leírni.

Pl: könyv, olvasó

[5] Computer Aided System Engineering

4.2.2 Tulajdonságtípus (attribútum)

Az attribútumok az egyedek jellemzõ jegyei.

Az attribútumok lehetnek:

Csoportosítás I.	**Csoportosítás II.**
egyszerûek	egyértékû
összetettek	többértékû

Kulcs attribútum: Olyan attribútum, amely egyértelmûen azonosítja az egyedtípus bármely elõfordulását.

Pl.: ISBN, cím, szerzõ stb. a könyv egyed esetében.

4.2.3 Típus és elõfordulás

A fenti absztrakciók esetén beszélünk egy adott típusú értékrõl mint elõfordulásáról.

4.2.4 Kapcsolattípus

Az egyedek logikai viszonya, összefüggése.

A kapcsolat lehet:

- **teljes**: a kapcsolatban lévõ egyedtípusok minden elõfordulására fennáll a kapcsolat
- **részleges (parciális)**. a kapcsolatban lévõ egyedtípusok nem minden elõfordulására áll fenn a kapcsolat.

A kapcsolatok a következõ típusúak lehetnek:

- **A két egyed független egymástól**
  Nincs kapcsolat a két egyed között.
- **A két egyed között kölcsönösen egyértelmû kapcsolat áll fenn, 1-1 kapcsolat**
  Egyik egyed egyedelõfordulásai a másik egyed legfeljebb egy egyedelõfordulásával létesítenek kapcsolatot Pl: házastárs
- **Egyik irányba egyértelmû, a másik irányba többértelmû a kapcsolat, 1-N kapcsolat**
  könyv - kiadó
- **N-M kapcsolat**
  Pl: **elõjegyzés**: olvasó - könyv esetén egy olvasó több könyvre is elõjegyezhet, és egy könyvre több olvasó is elõjegyezhet.
- **N-ágú kapcsolat**
  Pl: versenyez: a helyszín, idõpont és sportoló egyedek között. Kapcsolat fokszáma megadja, hogy a kapcsolatban hány egyed vesz részt.

4.2.5 Gyenge egyedtípus

Olyan egyedtípus, aminek nincs kulcs attribútuma, de van olyan vele kapcsolatban lévõ más egyedtípus (szülõ), amellyel való kapcsolata révén (azonosító kapcsolat) a gyenge egyedtípus egyedeit egyértelmûen azonosítani lehet

Pl: szülõ – gyerek; tulajdonos - hal

4.3 ER[6] (egyed kapcsolat) modell

Egy magas szintû, logikai adatmodell, amely **egyedtípus**okból, a köztük lévõ **kapcsolat**okból, és az egyes egyedtípusokhoz tartozó **attribútum**okból épül fel. Modellezéskor az adatbázis tervezõje dönti el, hogy mit kíván tulajdonságokkal (attribútumokkal), és mit új egyeddel leírni.

Megjegyzés:

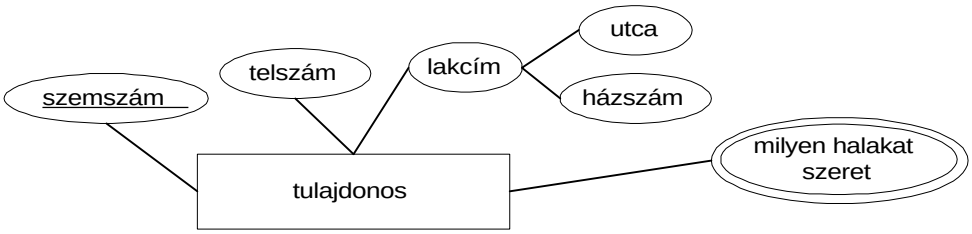
- Döntéseikor a minél kisebb rendundanciára és a minél gyorsabb adatelérésre törekszik, s közben az adatbázis mérete sem mellékes.
- Kiinduláskor jól kell specifikálni az egyedeket.

4.3.1 ER diagram komponensei

Egyedtípus és a gyenge egyedtípus ábrázolása:

Attribútumok ábrázolása:

kulcs, egyszerû, összetett és többértékû



Kapcsolattípusok ábrázolása:

1:N kapcsolat speciálisan gyenge egyedtípussal. A gyenge egyedtípus a hal, azonosító egyedtípusa a tulajdonos, N oldal teljes, 1 oldal parciális.

4.3.2 Példa ER modellre

Ha az a feladatunk, hogy egy nyilvántartást készítsünk, akkor elõször a feladat specifikációját kell megadni. Ezután kerülhet sor az ER modell megalkotására.

Feladat: Csokoládé nyilvántartás, Váthy Ágnes, Németh Krisztián: Adatmodellezési feladatok I. 45.old.

[6] Entity Relationship

4.4 A Suli-könyvtár ER modellje

4.4.1 Feladat specifikációja

Nyilván akarjuk tartani:

- a könyvtári könyveket (az egyes könyvek példányait)
- az olvasókat
- a példányok kölcsönzését
- az elõjegyzéseket könyvekre.

Össze kell gyûjteni a szükséges tulajdonságokat, és kapcsolatokat …

Az összetartozó tulajdonságok egyedet határoznak meg.

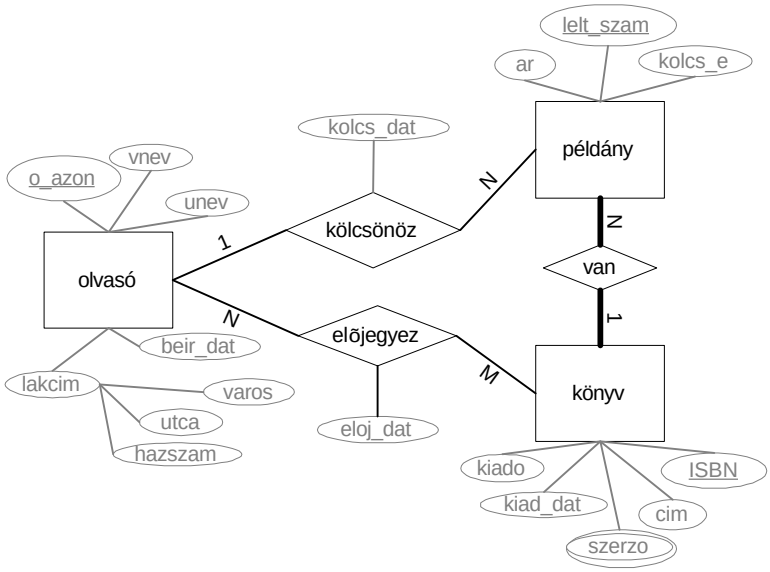
Induláskor legalább a következõ egyedek azonosíthatóak:

- olvasó
- példány
- könyv.

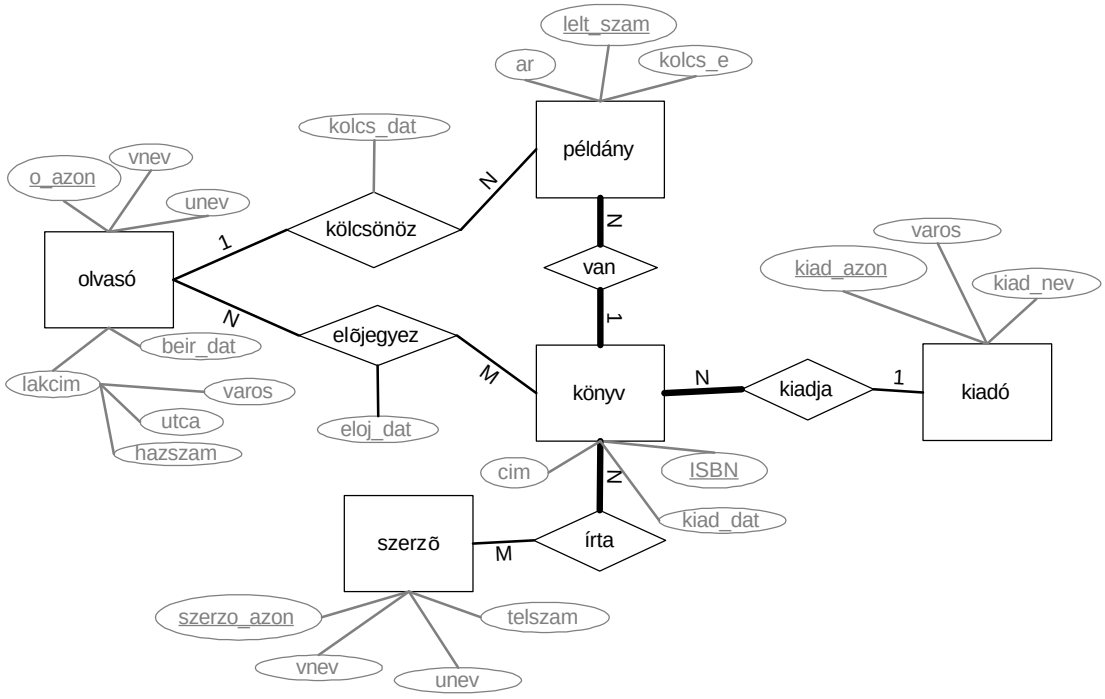
A kapcsolatok:

- kölcsönöz (olvasó példányt)
- elõjegyez (olvasó könyvre)
- van (könyvbõl példány).

Így az ER modell:



Abban az esetben, ha a szerzõkrõl és a kiadókról több adatot szeretnénk nyilvántartani, akkor a szerzõ és a kiadó az ER modell felrajzolása elõtt már egyednek tekintendõk



4.5 Hálós adatmodell

A hálós modell és a hozzá kapcsolódó adatbázis-kezelõ nyelv 1971-ben került a nyilvánosság elé a CODASYL DBTG[7] javaslata alapján. Hálós adatbázis-kezelõk : ADABAS, DBMS, IDMS (IBM), SÁMÁN

A modell alapvetõ elemei a rekord (az egyedtípus) és a halmaz (set).

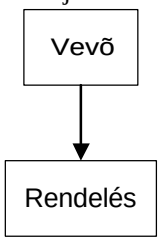
4.5.1 Settípus (halmaztípus), setelõfordulás

A modell alapegysége a settípus, ami két egyedtípus közötti kapcsolatot írja le.

A settípust a következõk alkotják:

- settípust neve (pl. Rendel)
- tulajdonos (owner) egyedtípus (pl. Vevõ)
- tag (member) egyedtípus (pl. Rendelés)

A set jelölése:



A rekordtípusban egyszerû, összetett és többértékû attribútumok is lehetnek, az utóbbiakat vektormezõben ábrázolják.

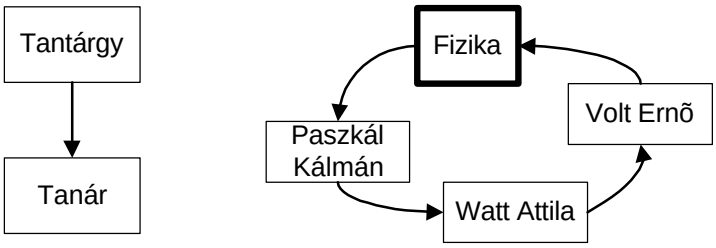
Egy settípus tulajdonképpen egy 1:N kapcsolatot (speciálisan 1:1 kapcsolatot) reprezentál.

Valamely settípus egy elõfordulása egy tulajdonos rekordból és az ahhoz kapcsolódóan valamennyi (0,1,2,…) tagrekordból áll. Egy settípusnak annyi elõfordulása van amennyi a tulajdonos rekordtípusnak.

[7] Conference on Data Systems Languages Data Base Task Group

Példa:

Tegyük fel, hogy a Tantárgy - Tanár egyedtípusok között 1:N a kapcsolat. Ekkor a kapcsolatot leíró settípus és annak egy elõfordulása:



Az egyes rekordelõfordulásokat mutatók láncolják egymáshoz (ciklikus lista).

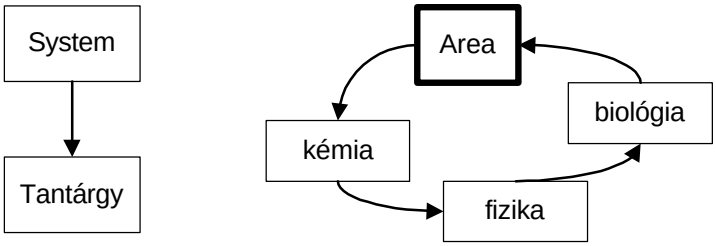
Mutatók:

- First - tulajdonostól az elsõ tagrekordra
- Next - tagtól a következõ tagra
- Prior - tagtól az elõzõ tagra
- owner - tagtól a tulajdonos rekordra

Ez a mutató szerkezet lehetõvé teszi, hogy egy adott setelõfordulás bármely adatelemét elérjük.

A Tantárgy rekordtípus hogyan tárolódik?

Ún. rendszertulajdonos set formájában, amelyben a tulajdonos ismeretlen a felhasználó számára. Az ilyen típusú setnek (singular set) csak egy elõfordulása van.

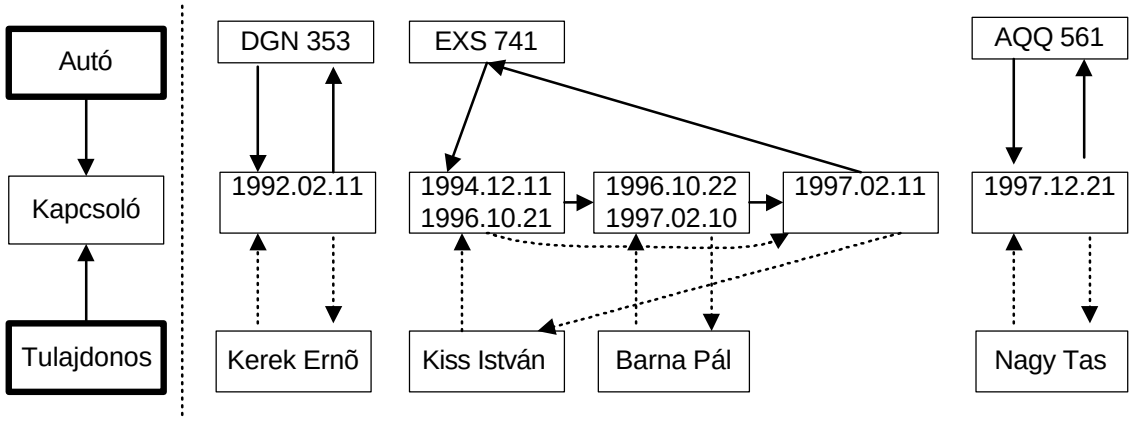


4.5.2 N:M kapcsolat

A hálós adatmodellben N:M kapcsolatok is kezelhetõk. Ekkor az egymással N:M kapcsolatban lévõ két rekordtípuson kívül egy ún. kapcsolórekordra is szükség van, ami segítségével az N:M kapcsolatot két 1:N kapcsolattá alakíthatjuk.

Példa

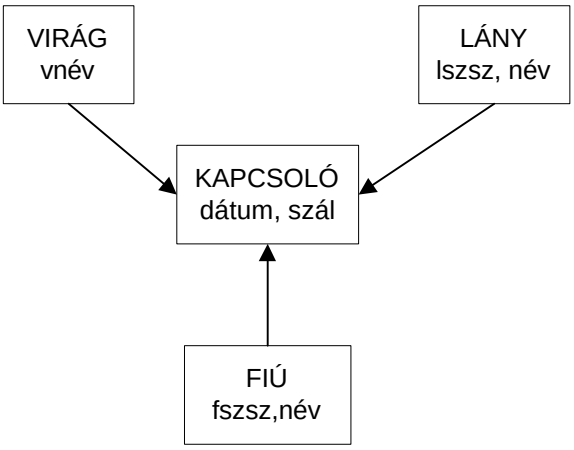
Tulajdonos és az Autó egyedek között N:M a kapcsolat, ha az autók egész életérõl nyilvántartást szeretnénk készíteni.



A kapcsolórekord minden egyes elõfordulása pontosan egy kapcsolatot létesít egy Autó és egy Tulajdonos rekord között, emellett még tárolja a mindkét egyedtõl együttesen függõ tulajdonságokat.

4.5.3 N-ágú kapcsolat

Az N-ágú kapcsolat hálós adatmodellbeli realizációja a következõ:



4.5.4 CODASYL DBTG javaslat

1969 elsõ változat, 1971-ben végleges jelentés, amelynek célja olyan rendszer definiálása volt, amely a következõ tulajdonságokkal rendelkezik:

- különbözõ felhasználók és programok különbözõ adatokat igényelhessenek, redundancia nélkül
- lehetõvé teszi adatstruktúrák definiálását
- az adatbázis és leírása különbözõ programnyelvekhez illeszthetõ legyen.
- lehetõvé teszi, hogy a programok függetlenek legyenek az adattárolóktól
- lehetõvé teszi, hogy a programok függetlenek legyenek az adatoktól
- több felhasználó ill. program is hozzáférhessen egyidejûleg az adatbázishoz
- adatvédelem

E célok elérésére három nyelvre tettek javaslatot:

- Sémaleíró nyelv: leírja az adatmodellt és a fizikai tárolás módját. Egy séma leírásakor meg kell adnunk a séma nevét, területek (area) leírását, rekordok leírását, set-ek leírását.
- Alsémaleíró nyelv: kapcsolatot létesít a séma és a felhasználói program között, ezért szintaxisának összhangban kell lennie azokkal a programnyelvekkel, amelyekbõl a dbms hívható (az akkor javasolt nyelv a COBOL-hoz illeszkedik). Alséma leírásakor a dba rögzíti, hogy az alsémával dolgozó program a teljes adatbázisnak mely részéhez férhet hozzá.
- Adatkezelõ nyelv: adatok manipulációját végzi (beszúrás, törlés, módosítás, rendezés, keresés stb.)

4.5.5 Leképezés ER modellrõl hálós modellre

Az ER modellbõl hálós adatmodellt hozhatunk létre szinte automatikusan az ún. leképezési szabályok alkalmazásával.

ER modellben	**Hálós modellben**
egyedtípus	rekordtípus
többértékû attribútum	vektormezõ
összetett attribútum	csoport
gyenge egyedtípus	a) az azonosító egyedtípusnak megfelelõ rekordtípust egy vektormezõvel egészítjük ki, amely tartalmazza a gyenge egyedtípus attribútumait (ha a gyenge egyedtípus nem szerepel további kapcsolatokban) b) létrehozunk egy settípust, melynek tulajdonosa az azonosító rekordtípus, tagja a gyenge egyedtípus
1:1 kapcsolattípus	settípus (tag a teljes oldal, tulajdonos parciális oldal)
1:N kapcsolattípus	settípus (tag az N oldal, tulajdonos 1 oldal)
bináris M:N kapcsolattípus	kapcsoló rekordtípus születik
többágú M:N kapcsolattípus	kapcsoló rekordtípus születik, amely annyi setnek lesz tagja, ahány ágú a kapcsolat

4.6 Hierarchikus adatmodell

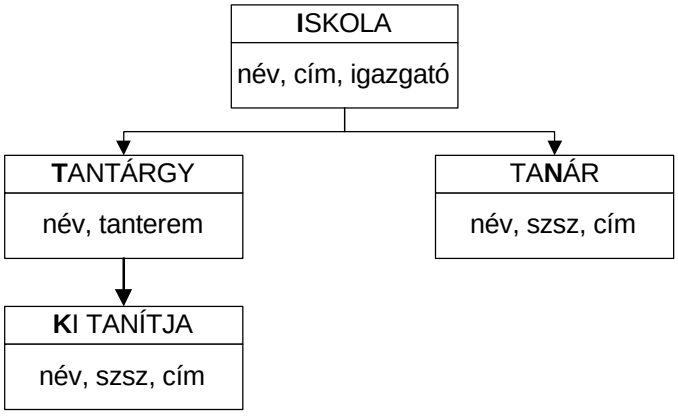
Az elsõ hierarchikus adatbázis-kezelõ program az IBM által kifejlesztett Information Management System (IMS) volt 1968-ban.

4.6.1 Alapfogalmak

rekordtípus = egyedtípus

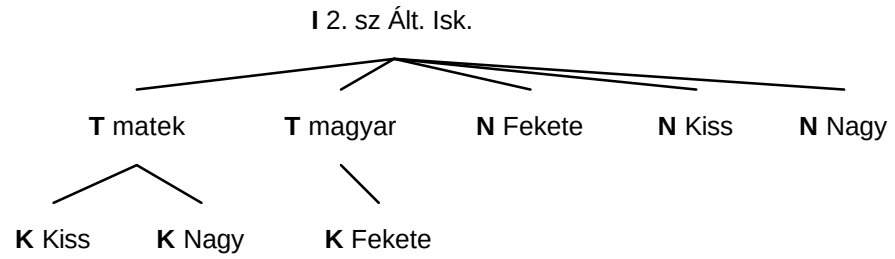
szülõ-gyerek kapcsolattípus = 1:N kapcsolattípus
(Parent - Child Relationship/PCR)

Ábrázolás: hierarchiadiagrammal



Megjegyzés: A PCR kapcsolatoknak nincs nevük.

A fenti hierarchiadiagram egy efõfordulásdiagramja egy ún. elõfordulási fa (rekordtípus=csomópont, PCR kapcsolattípus=ág):



Egy elõfordulási fát a fa inorder bejárása alapján tároljuk.

4.6.2 A klasszikus hierarchikus modell tulajdonságai

1. Egyetlen rekordtípus a gyökér nem vehet részt PCR kapcsolatban
2. Minden rekordtípus gyerekként pontosan egy PCR kapcsolatban vehet részt.
3. Bármely rekordtípus szülõként tetszõleges számú PCR kapcsolatban részt vehet.

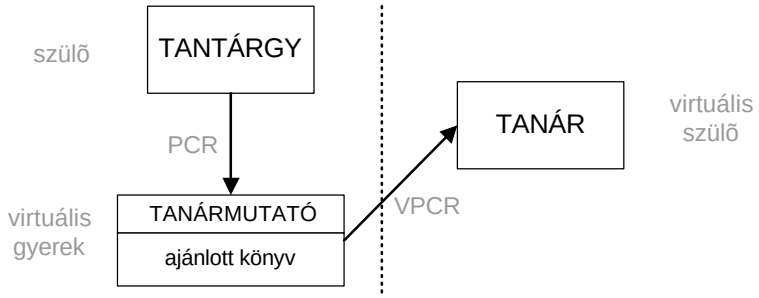
Megjegyzés: A fentiek miatt a klasszikus hierarchikus modellben csak 1:1 és 1:N kapcsolatok valósíthatók meg.

4.6.3 Az N:M kapcsolatok megvalósítása

- a gyerek rekordok ismétlõdõ elõfordulása → nagy redundancia
- virtuális szülõ - gyerek kapcsolat (VPCR[8])

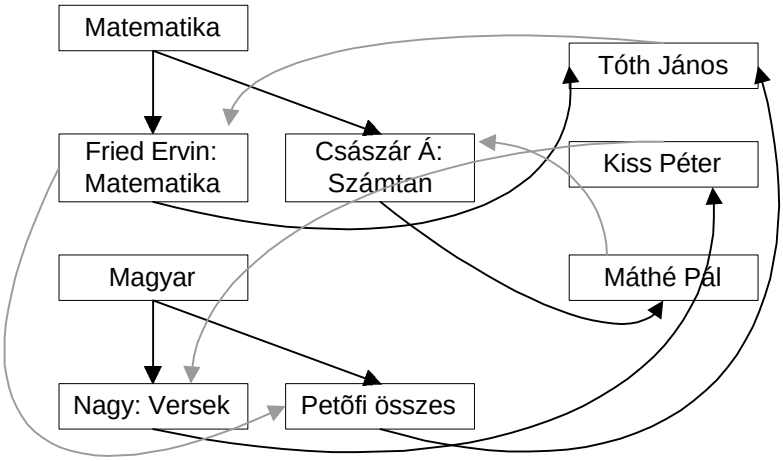
A VPCR megvalósításához egy virtuális (mutató) rekordtípust kellett bevezetni, aminek az M:N kapcsolat megvalósításán túl az is a feladata, hogy mindkét rekordtípusra jellemzõ attribútumot tároljon.

Ha a Tanár és a Tantárgy rekordtípus között N:M a kapcsolat akkor a modellt a következõképpen ábrázoljuk a virtuális (mutató) rekordtípus használatával:



[8] Virtual Parent Child Relationship

A fenti modell néhány elõfordulása a következõképpen néz ki:



Megjegyzés: Ez a tárolási struktúra hasonlít a hálós modellbeli megvalósításra. A hierarchikus adatmodell az olyan feladatok megvalósítására, amelyekben sok a nem hierarchikus N:M kapcsolat nem alkalmas.

4.6.4 Leképezés ER modellrõl a hierarchikus modellre

Az ER modellrõl a hierarchikus modellre történõ átalakítást az ún. leképezési szabályok teszik automatikussá.

ER modellben	Hierarchikus modellben
egyedtípus	rekordtípus
1:N kapcsolat	PCR kapcsolat
N:M kapcsolat	VPCR kapcsolat
N ágú kapcsolat	VPCR kapcsolat

4.7 Relációs adatmodell

Ez a modell van kidolgozva a legrészletesebben, ami E.F. CODD-nak köszönhetõ, aki a hetvenes évek elején dolgozta ki. Ebben a modellben valósítható meg legjobban a fizikai és logikai adatfüggetlenség.

4.7.1 Reláció fogalma

Def: Legyenek $D_1,..,D_n$ attribútumhalmazok. Az $R \subseteq D_1 \times ... \times D_n$ halmazt relációnak nevezzük, ahol *n* a reláció foka.

A reláció felfogható táblázatként. Ha a D_i elemeit d_i^j -vel jelöljük (i=1,…,n), akkor

D_1	…	D_n
d_1^1		d_n^1
d_1^m		d_n^m

A reláció mint halmaz elemeit (a táblázat sorait) rekordoknak is nevezzük.

Az attribútumhalmazok neveit (oszlopfejléceket) mezõnek is mondjuk.

Megjegyzés:

A halmazelméleti megfontolások miatt a táblázatban

- a sorok sorrendje lényegtelen,
- az oszlopok sorrendje is lényegtelen,
- nincs két azonos sor.

Egy adatbázis rendszerint több táblából áll, a táblák között kapcsolatok vannak. A táblák struktúrájának illetve a kapcsolatoknak a leírását az adatszótár tartalmazza.

Magas szintû és a relációs adatmodell fogalmai a következõképpen felelnek meg egymásnak:

Magas szintû adatmodell fogalmai	Relációs adatmodell
egyed	reláció
egyedelõfordulás	rekord
attribútum/egyedtípus	mezõ

4.7.2 Kulcsok, funkcionális függõség

Def (kulcs): $K \subseteq \{D_1,...,D_n\}$ kulcs, ha $\forall$ s1,s2 sor esetén, ha s1|K=s2|K, akkor s1=s2, és ha $\forall$ K` $\subseteq$ K esetén teljesül a fenti tulajdonság, akkor K`=K teljesül. Ha az utolsó feltétel nem teljesül, akkor K-t szuperkulcsnak is mondják. Ha a kulcs egy attribútumhalmaz, akkor *egyszerû*nek, egyébként *összetett*nek mondjuk.

Példa:

Az ELOJEGY táblában az ISBN és az o_azon együtt alkotják a kulcsot, hiszen együttesen egyértelmûen meghatározzák az elõjegyzés dátumát (eloj_dat), ez önmagában sem az ISBN, sem pedig az o_azon mezõre nem teljesül. (Egy könyvre több különbözõ olvasó, más-más idõpontokban jegyezhetett elõ; és egy olvasó több könyvre is elõjegyezhet egy adott idõpontban.)

Def (külsõ vagy idegen kulcs): Egy reláció azon attribútumai, amelyek egy másik relációban kulcsot alkotnak.

Példa:

Az ELOJEGY táblában az ISBN és az o_azon mezõk egyenként külsõ kulcsok, hiszen csak olyan ISBN szám szerepelhet a táblában, amely a KONYV táblában is létezik (az ISBN a KONYV táblában kulcs), és csak olyan

o_azon érték szerepelhet a táblában, ami az OLVASO táblában is megvan (az o_azon az OLVASO táblában kulcs).

4.7.3 Normalizálás

Egy adatbázis relációs adatmodelljének (koncepcionális modell) megalkotásához két út vezet

- leképezési szabályok alkalmazásával ER modellbõl relációs modell (lásd késõbb)
- a feladat specifikációjából (tárolandó adatok, mûveleti igények) kiindulva normalizálással jutunk el az adatbázis relációs adatmodelljéhez.

A normalizálással:

- csökken a redundancia
- megszûnnek a törlési, módosítási, beszúrási anomáliák
- logikailag áttekinthetõbb lesz az adatbázis

Def (funkcionális függ.): Legyen R a $D_1,..,D_n$ attribútumhalmazokon értelmezett reláció, és legyenek A_i és $A_j \subset \{D_1,\ldots,D_n\}$. Azt mondjuk, hogy az R relációban A_j funkcionálisan függ A_i-tõl ($A_i \rightarrow A_j$), ha $\forall$ s1, s2 $\in R$ esetén abból, hogy $s1(D_i) = s2(D_i)$ $\forall D_i \in A_i$-re következik, hogy $s1(D_j) = s2(D_j)$ $\forall D_j \in A_j$-re.

Def (teljes funkcionális függõség): Ha A→B és ha $\forall$ A` része A esetén A'→B, akkor A`=A.

Def (tranzitív függõség): C tranzitívan függ A-tól, ha létezik B része {D1,...,Dn}, hogy A→B és B→C és B-/->A és C-/->B.

4.7.3.1 Elsı normÆlforma

Def: Egy reláció **elsõ normálformájú**, ha az értelmezési tartományának egyetlen eleme sem reláció, azaz ha a táblázat minden cellájában csak egy attribútumérték szerepel.

Példa:

Legyen a relációnk a következõ: R(A, B, C, D,E). A kulcsjellegû tulajdonságok alá vannak húzva.

A	**B**	**C**	**D**	**E**
a1	b1	c1	d1	e1
		c2	d2	e2
		c3	d3	e3
a2	b1	c2	d2	e2

A fenti reláció nem elsõ normálformájú, mert ismétlõdõ csoportot tartalmaz (a C,D,E attribútumok értékei).

4.7.3.2 RelÆciÓ elsı normÆlformÆra hozÆsa

4.7.3.2.1 Sorok ismétlésével

A több attribútumértéket tartalmazó sort annyi sorra bontjuk ahány attribútumérték fordul elõ a sorban → redundancia.

A	**B**	**C**	**D**	**E**
a1	b1	c1	d1	e1
a1	b1	c2	d2	e2
a1	b1	c3	d3	e3
a2	b1	c2	d2	e2

4.7.3.2.2 Reláció felbontásával

A reláció újabb relációkra bontható úgy, hogy az ismétlõdõ csoportot leválasztjuk az eredeti relációról, melléjük illesztve a nem ismétlõdõ rész kulcsát.

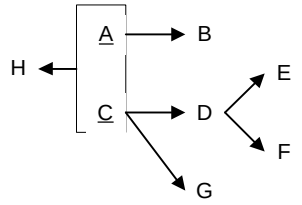
A	**B**
a1	b1

A	**C**	**D**	**E**
a1	c1	d1	e1
a1	c2	d2	e2
a1	c3	d3	e3
a2	c2	d2	e2

4.7.3.3 MÆsodik normÆlforma

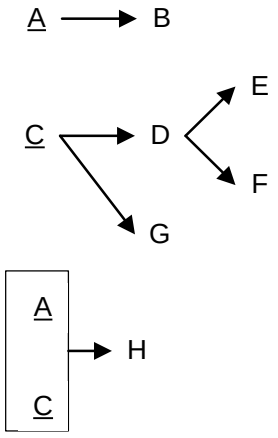
Def: Egy reláció **második normálformájú**, ha 1NF-jú és minden olyan attribútum, ami nem kulcs teljesen funkcionálisan függ minden kulcstól.

Tekintsük az alábbi 1NF-jú relációt. Az A, B, C, D, E, F, G, H, attribútumok (tulajdonságtípusok), a relációban, az A és B tulajdonságok a kulcsok. A reláció nem második normálformájú, mert vannak olyan másodlagos attribútumok, amelyek a kulcs részeitõl is függnek.



4.7.3.4 RelÆciÓ mÆsodik normÆlformÆra hozÆsa

Ha egy elsõ normálformájú relációban a kulcs egyszerû, akkor a reláció egyben második normálformájú is. Egyébként az összetett kulcsú relációban meg kell vizsgálni azokat az attribútumokat, amelyek nem részei a kulcsnak. Ha ezek között az ún. másodlagos attribútumok között vannak olyanok, amelyek nem függnek teljesen funkcionálisan a kulcstól (azaz nincsen a reláció a második normálformában), akkor meg kell határozni, hogy ezek a tulajdonságok mely részkulcstól függnek teljesen, és a tulajdonságokat a részkulccsal együtt külön táblázatba kell tenni úgy, hogy ott a részkulcs már kulcs legyen.



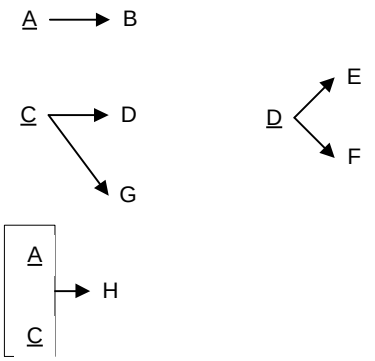
4.7.3.5 Harmadik normÆlforma

Def: Egy reláció **harmadik normálformájú**, ha 2NF és nincs olyan másodlagos attribútum, ami tranzitív módon függne valamilyen kulcstól.

4.7.3.6 RelÆciÓ harmadik normÆlformÆra hozÆsa

A tranzitív függõségeket úgy tüntetjük el, hogy azokat külön táblázatba vagy táblázatokba tesszük.

A fent eredményül kapott relációk közül a középsõ nincs harmadik normálformában, hiszen az E és F attribútumok tranzitívan függnek a C kulcstól. A harmadik normálformára hozáshoz küszöböljük ki a középsõ relációból a tranzitív függõséget.



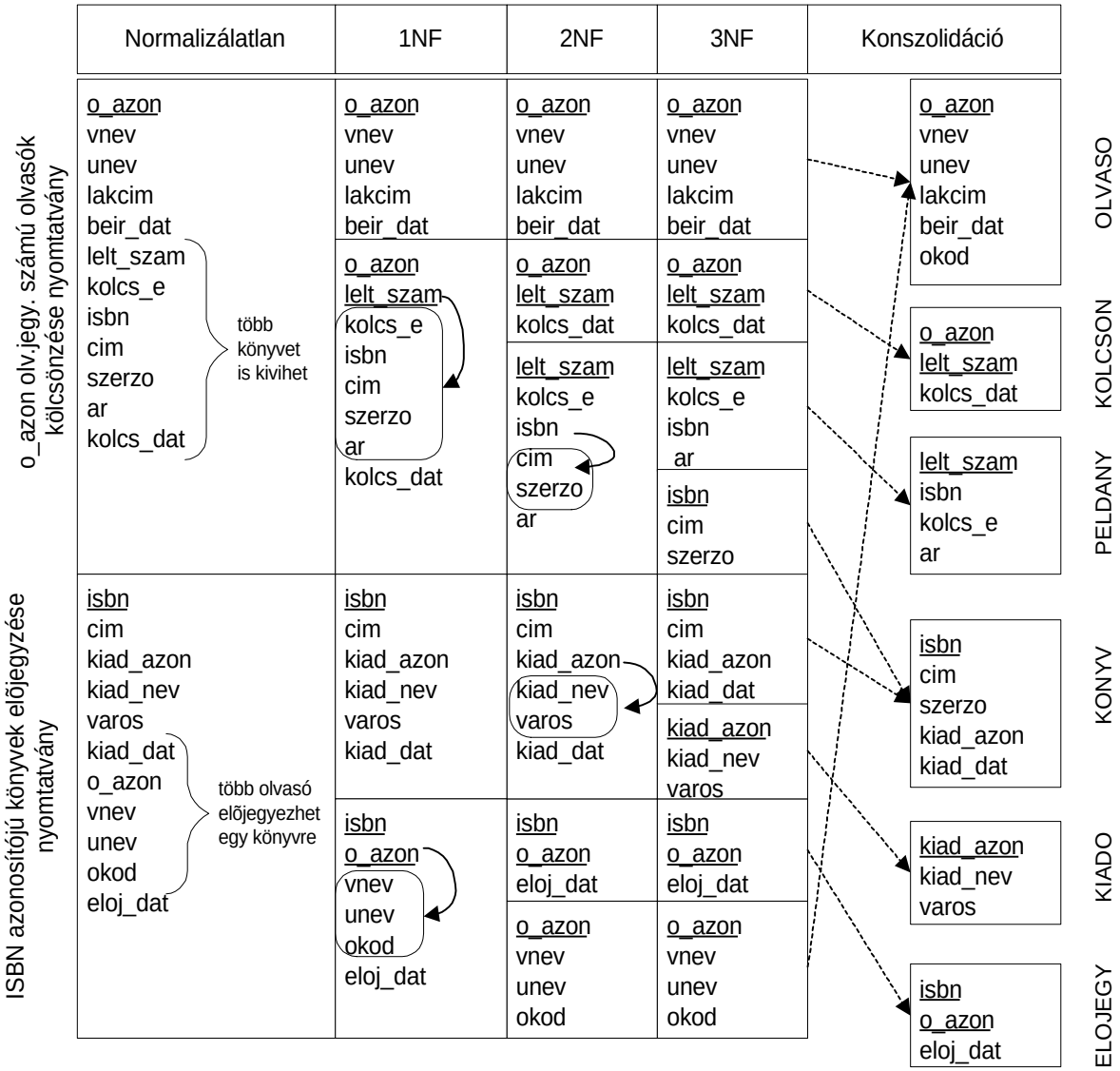
Az eredményül kapott négy reláció mindegyike harmadik normálformában van.

4.8 A Suli-könyvtár relációs adatbázisának megtervezése normalizálással

A munka menete:

1. Feladat specifikációja (tárolandó adatok, mûveleti igények összegyûjtése)
2. kiinduló relációk elkészítése
3. normalizálás
4. konszolidáció

Az adatbázisunk a relációs adatmodellen fog alapulni.



4.9 Leképezési szabályok, áttérés ER modellrõl relációs modellre

Az ER modellrõl leképezési szabályok használatával hozhatjuk létre az alacsonyabb szintû logikai adatmodelleket (hálós, hierarchikus, relációs). A leképezési szabályok alkalmazása automatizálható (CASE[9] TOOLS).

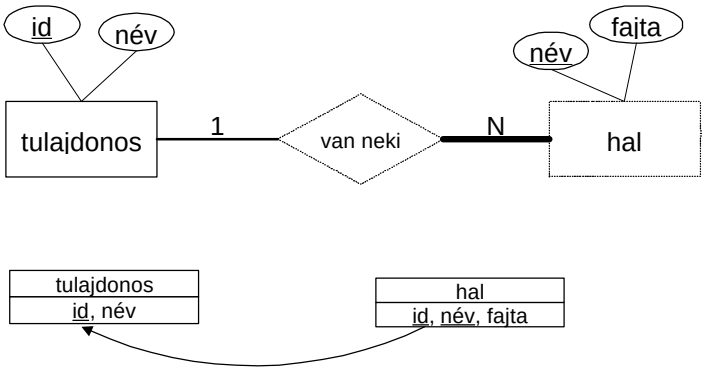
4.9.1 Egyedtípus leképezése

Minden egyes egyedtípusnak (a gyenge egyed kivételével) egy relációt feleltetünk meg. Az egyedtípus attribútumai lesznek a reláció mezõi: a kulcsattribútumok az elõdleges kulcsok, az összetett attribútumokat komponenseikre kell felbontani.

4.9.2 Gyenge egyed leképezése

A gyenge egyedtípusnak olyan relációt feleltetünk meg, amelyben a gyenge egyed összes attribútuma mellett a szülõ (azonosító) egyedek kulcsai szerepelnek, és ezek a kulcsok a gyenge egyed parciális kulcsával (ha van) alkotják a reláció elsõdleges kulcsát. Az azonosító egyedek kulcsai a létrejött relációban külsõ kulcsok lesznek.

Példa:



4.9.3 1:1 kapcsolattípus leképezése

Esetek:

- totális E1 és totális E2
- parciális E1 vagy E2
- parciális E1 és E2

4.9.3.1 TotÆlis E1 Øs totÆlis E2

Összevonjuk a két egyedet egy egyedtípusba, a két kulcs közül az egyiket hagyjuk meg.

Ha valamelyik egyed más kapcsolatban is részt vesz, akkor érdemes meghagyni mindkét egyedet relációnak úgy, hogy az egyik relációt kiegészítjük a másik reláció kulcsával mint idegen kulccsal.

Példa: Az osztály alkalmazottai egyik irodában alk_azon, a másik osztályon szig_szám szerint vannak nyilvántartva.

4.9.3.2 ParciÆlis E1 Øs totÆlis E2

Az egyes egyedekbõl reláció lesz úgy, hogy a teljes kapcsolatban lévõ relációhoz hozzávesszük a részleges reláció kulcsát idegen kulcsként.

Példa: családfõ – házastárs

[9] Computer Aided Software Engineering

4.9.3.3 ParciÆlis E1 Øs parciÆlis E2

Az egyes egyedtípusokból reláció lesz, valamint a kapcsolattípusból is célszerû relációt készíteni (az üres mezõk ún. NULL értékek elkerülése végett) úgy hogy az utóbbi relációhoz hozzávesszük a két egyedhez tartozó relációk kulcsait.

Példa: Nõ - *házastárs* - férfi

4.9.4 1:N kapcsolat leképezése

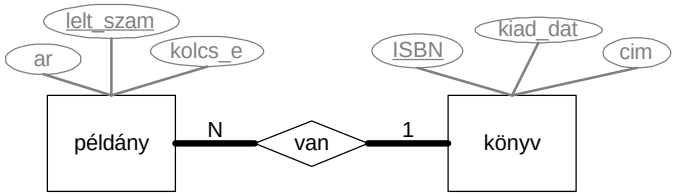
Esetek:

- totális N oldal
- parciális N oldal

4.9.4.1 totÆlis N oldal

Az egyes egyedekbõl relációk lesznek úgy, hogy az N oldali relációt kiegészítjük az 1 oldali reláció kulcsával mint idegen kulccsal.

Példa



könyv - *van* - példány

könyv(ISBN, cim, kiad_dat), példány (lelt_szam, ISBN, kolcs_e, ar)

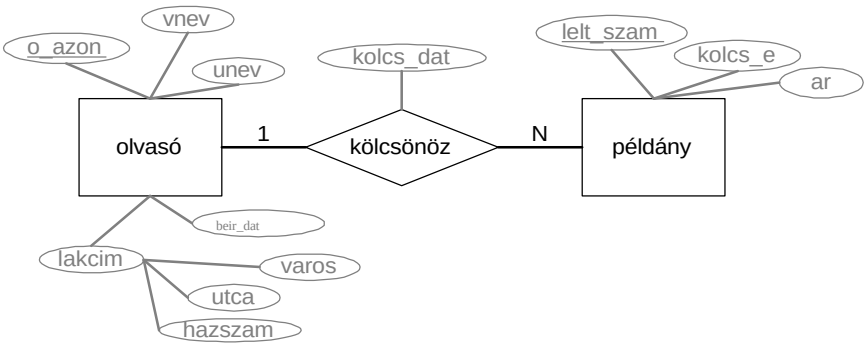
könyv - *kiadja* - kiadó

kiadó(kiad_azon, kiad_nev, varos), könyv(ISBN, cim, kiad_dat, kiad_azon)

4.9.4.2 parciÆlis N oldal

Az egyes egyedekbõl relációk lesznek, és a kapcsolatból is célszerû relációt készíteni (az üres mezõk ún. NULL értékek elkerülése végett). A kapcsolatból származó reláció az egyedek kulcsait és ha a kapcsolatnak vannak attribútumai, akkor azokat tartalmazza. A kapcsolatból származó reláció kulcsa az N oldali reláció kulcsa lesz.

Példa



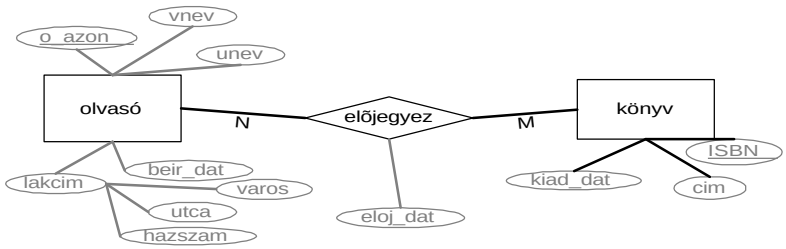
olvasó - *kölcsönöz* - példány

olvasó(o_azon, vnev, unev, varos, utca, hazszam, beir_dat), példány(lelt_szam, kolcs_e, ar), kölcsönöz(lelt_szam, o_azon, kolcs_dat)

4.9.5 N:M kapcsolat, n-ágú kapcsolat leképezése

A leképezési szabály részvételtõl függetlenül az, hogy az egyedekbõl és a kapcsolatból reláció lesz. A kapcsolat reláció az egyes egyedek kulcsaiból (együttesen lesznek ezen reláció kulcsa) és a kapcsolat attribútumaiból áll.

Példa



olvasó - *elõjegyez* - könyv

olvasó(o_azon, , vnev, unev, varos, utca, hazszam, beir_dat), könyv(ISBN, cim, kiad_dat), elõjegyez(o_azon, ISBN, eloj_dat)

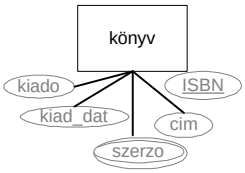
szerzõ –*írta* -könyv

szerzõ(szerzo_azon, vnev, unev, telszam), könyv(ISBN, cim, kiad_dat), írta(szerzo_azon, ISBN)

4.9.6 Többértékû attribútum leképezése

A többértékû attribútumot megszüntetjük, új relációt hozunk létre belõle úgy, hogy ehhez a relációhoz hozzávesszük a többértékû attribútum egyedének kulcsát.

Példa



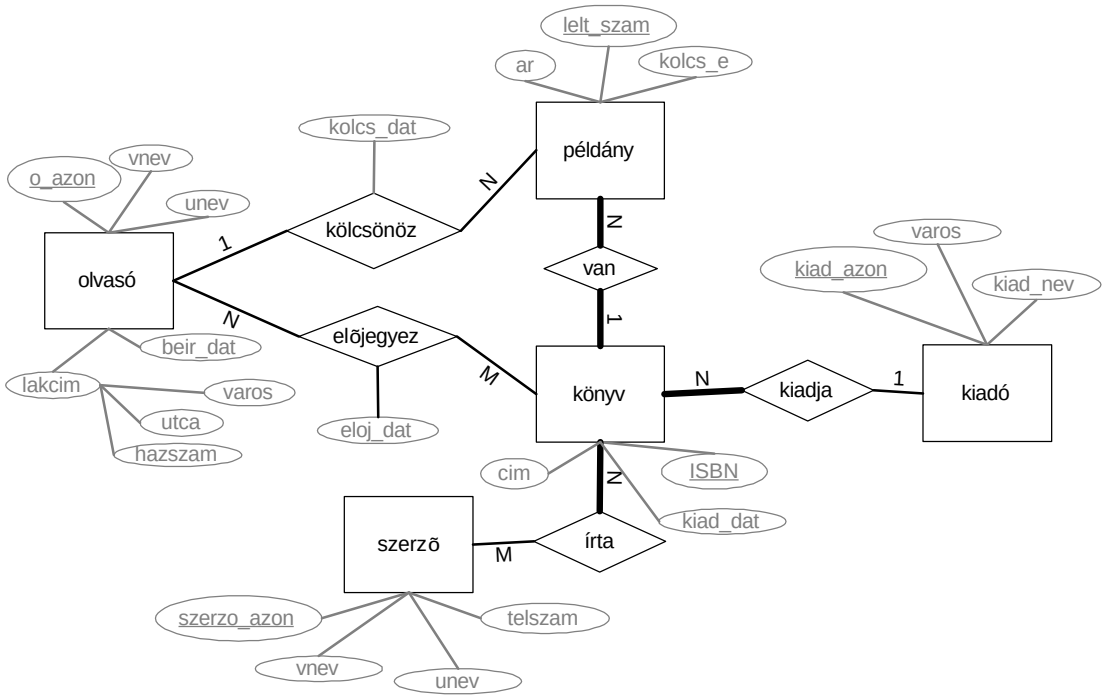
Megjegyzés: Ha a szerzõvel kapcsolatban a címét, telefonszámát stb. is nyilván akarjuk tartani, akkor célszerû már az ER modell megalkotásakor külön egyednek tekinteni, és N:M kapcsolatot feltételezni a könyv egyeddel.

4.10 A Suli-könyvtár relációs adatbázisának megtervezése ER modellbõl

A munka menete:

1. Feladat specifikációja (tárolandó adatok, mûveleti igények összegyûjtése).
2. ER modell elkészítése.
3. Leképezési szabályok alkalmazásával relációk létrehozása.
4. Konszolidálás.
5. Normalizálás, ha további finomítás szükséges.

4.4 részben elkészítettük a következõ ER modellt:



Alkalmazzuk a leképezési szabályokat!

könyv - *van* - példány

könyv(ISBN, cim, kiad_dat), példány (lelt_szam, ISBN, kolcs_e, ar)

könyv - *kiadja* - kiadó

kiadó(kiad_azon, kiad_nev, varos), könyv(ISBN, cim, kiad_dat, kiad_azon)

olvasó - *elõjegyez* - könyv

olvasó(o_azon, , vnev, unev, varos, utca, hazszam, beir_dat), könyv(ISBN, cim, kiad_dat), elõjegyez(o_azon, ISBN, eloj_dat)

szerzõ –*írta* -könyv

szerzõ(szerzo_azon, vnev, unev, telszam), könyv(ISBN, cim, kiad_dat), írta(szerzo_azon, ISBN)

olvasó - *kölcsönöz* - példány

olvasó(o_azon, vnev, unev, varos, utca, hazszam, beir_dat), példány(lelt_szam, kolcs_e, ar), kölcsönöz(lelt_szam, o_azon, kolcs_dat)

olvasó - *elõjegyez* - könyv

olvasó(o_azon, , vnev, unev, varos, utca, hazszam, beir_dat), könyv(ISBN, cim, kiad_dat), elõjegyez(o_azon, ISBN, eloj_dat)

szerzõ –*írta* -könyv

szerzõ(szerzo_azon, vnev, unev, telszam), könyv(ISBN, cim, kiad_dat), írta(szerzo_azon, ISBN)

Az azonos relációk uniójának vétele (konszolidálás).

Normalizálás, ha szükséges.

5. Fizikai tárolási, elérési mód

Ha adatbázis-kezelésrõl beszélünk, akkor az adatok tárolása közvetlen elérésû háttértárolón, általában mágneslemezen történik. Az itt tárgyalandó fogalmak az adatbázis-kezelõ rendszer kiválasztása után válnak konkréttá.

lemezterület ← fordított arányosság → gyors elérés

⇓

optimum=kompromisszum

Az adatszerkezetek fizikai tároláshoz szükséges lemezmennyiség nagysága és az adatok elérésének ideje között fordított arányosság figyelhetõ meg, azaz minél gyorsabbra tervezzük az adatok elérését, annál több dolgot kell tárolnunk az adatokkal kapcsolatban, így annál nagyobb lesz az adatbázis mérete. (Pl. sok mezõ szerint létrehozunk indexállományokat, akkor ezen állományok mérete növeli az adatbázis méretét.)

5.1 Mágneslemez

Fizikai jellemzõk

- lemezek, olvasófejek (oldalanként legalább egy)
- sáv, cilinder, szektor, blokk

Blokk, fizikai tárolási egység:

- az az adatmennyiség, amely együtt mozog a háttértár (mágneslemez) és az operatív tár (puffer) között íráskor ill. olvasáskor
- címezhetõ
- fizikai rekordokból áll

Hozzáférési idõt meghatározza a(z):

1. Fejmozgatás
2. Fejkiválasztás
3. Forgatás
4. Adatátvitel

Fizikai tárolásnál figyelembe veendõ szempontok

1. A várhatóan egyidejûleg vagy egymás után használt rekordok egy blokkon legyenek.
2. Kiíráskor a majdan egymás után beolvasandó rekordokat, egymás utáni blokkokra írjuk ki.
3. Az egyes rekordok hosszának minimalizálása. (Pl. cím mezõ 50 byte-nak van deklarálva, de csak 20 byte-ot használunk egy rekordban, akkor 30 byte felesleges. Megoldás: a cím mezõt, így a rekordot is változó hosszúságúnak deklaráljuk. Ekkor a rekord fizikai tárolásánál a mezõ elõtt lesz 2 byte, ami a mezõ aktuális hosszát tárolja.)

5.2 Szervezési módok és elérések

Az adatok szervezési és elérési módja lehet:

- soros (fizikai egymásutániság szerinti)
- szekvenciális (vmilyen logikai sorrend szerinti)
- direkt (indexelt)
- direkt (hashing)

5.2.1 Soros

Az adatok úgy helyezkednek el egymás után, hogy

- sem a kulcsok és a tárolón elfoglalt helyük között,
- sem az egymást követõ adatok kulcsai vagy egyéb tulajdonságai között

nincs semmilyen összefüggés.

Az adatok feldolgozása:

- sorban
- beszúrás a végére, vagy rendezett beszúráskor sok adatot kell mozgatni
- módosítás sok adat mozgatását igényelheti
- törlés sok adat mozgatását igényeli
- egy rekord megtalálása lassú (ha kulcsmezõ alapján keresünk, akkor átlagosan a rekordok felét, ha egyéb mezõ szerint akkor az összes rekordot végig kell keresni)

Rendszerint archiválni szoktak soros szervezésû és elérésû tárolóra (pl. mágnesszalag).

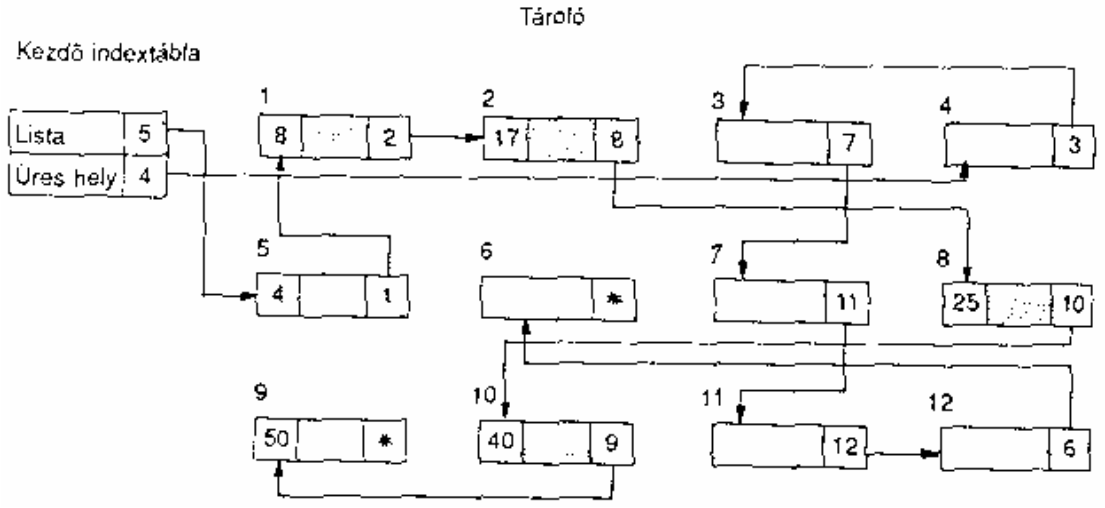
5.2.2 Szekvenciális

Szekvenciális szervezésnél az egymást követõ adatok kulcsa között logikai kapcsolat van. Pl. növekvõ sorrendnél minden rekord kulcsa nagyobb (vagy egyenlõ) a közvetlenül elõtte lévõjénél.

Szekvenciális elérés:

- bármely adat után csak a közvetlenül utána következõt érhetjük el
- ugyanazokon az adatokon többféle mezõ alapján is létrehozhatunk szekvenciális elérést
- az adatok között a logikai kapcsolatot mutatók határozzák meg, és ezek teszik lehetõvé a szekvenciális feldolgozást is. Szoktak még gyûrût és/vagy kétirányú láncolt listát is használni.

Ábra 4 Szekvenciális szervezés listaszerkezettel



A listaszerkezet elõnyei a soros szervezéssel szemben:

- a módosítás, beszúrás egyszerûbb
- ugyanazon rekordtípuson többféle szekvenciális szerkezet létrehozható
- egy adott kulcs átlagosan az összes rekord felének végignézése után megtalálható.

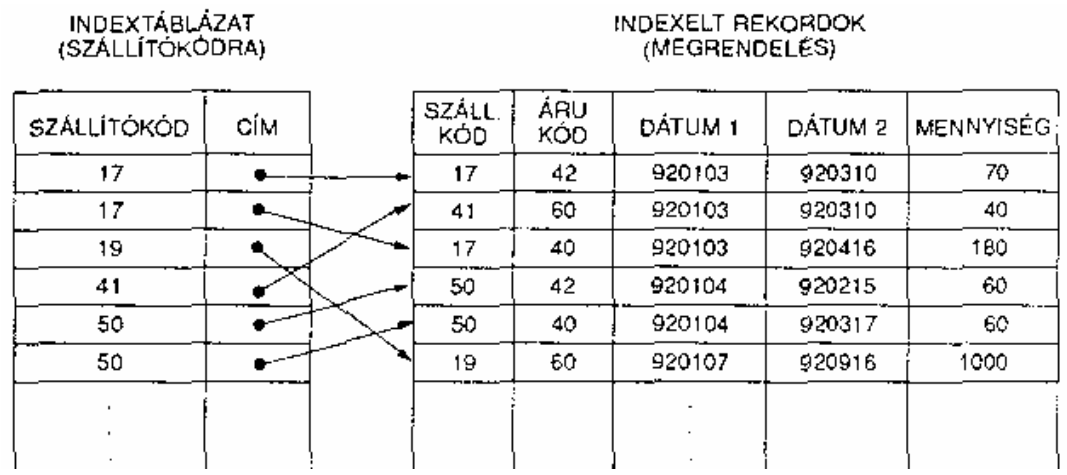
A listaszerkezet hátrányai:

- bonyolult programmal valósítható meg
- a mutatók miatt nagyobb a tárolóigénye.

5.2.3 Indexelés

Indexelni a relációs adatmodellben táblázatokat/relációkat szoktak valamilyen mezõ(k) alapján. Ahhoz hogy indexelni lehessen közvetlen elérésû háttértárra van szükség. Az indexelés eredményeképpen létrejön egy a táblázathoz tartozó indexfájl (indextábla), ami a következõképpen nézhet ki:

Ábra 5 MEGRENDELÉS rekordok indexelése SZÁLLÍTÓKÓD szerint



Indexelni lehet elsõdleges kulcs alapján → elsõdleges index, vagy egyéb mezõ alapján → másodlagos index. Így egy táblázathoz több indexfájl tartozhat.

A másodlagos indexfájl felépítése

- annyi sora van az indextáblának, ahány értéke van annak a mezõnek ami szerint indexeltük a táblázatot
- annyi sora van az indextáblának, ahány különbözõ értéke van annak a mezõnek ami szerint indexeltük a táblázatot.

A második esetben az egyes mezõértékek melletti cellában egy lista kezdõcíme van, ami ezen mezõértékhez tartozó rekordok fizikai címeit tárolja. Ez csökkenti az indextábla méretét, de a feldolgozást nehezíti.

Az indexelés elõnyei a szekvenciális szervezéssel és eléréssel szemben:

Egy rekord megtalálása olyan mezõérték alapján, amelyre létrehoztunk indexet az indexállományban történik. Ez sokkal kisebb állomány, mint a tábla, így gyorsabban lehet benne keresni (kevesebb blokkot kell beolvasni). Az adott mezõérték megtalálásakor az adott mezõérték mellet ott van az indextáblában az az információ is, hogy az adott mezõértékû rekord(ok) hol (melyik blokkban) találhatók a lemezen.

További információk Quittner Pál: Adatbázis-kezelés a gyakorlatban, Akadémiai kiadó, Budapest, 1993, 81.oldal

5.2.4 Hashing

Az indexek alapján történõ elérés viszonylag nagy adminisztrációt jelent az adatbázis-kezelõ rendszer számára, mivel:

- az indexállományokat létre kell hozni,
- karban kell tartani, hogy aktuális legyen adatmanipuláció után is, és
- keresni kell bennük.

A hashing módszer akkor használható jól, ha kulcs alapján akarunk adatokat **közvetlenül** elérni. Ennél a módszernél a kulcs és a megfelelõ adat címe között majdnem kölcsönösen egyértelmû megfeleltetés van. Egy

alkalmas ún. hash leképezés különbözõ kulcsértékhez majdnem mindig különbözõ címet (címtartományt) rendel. Bizonyos esetekben azonban különbözõ kulcsértékekhez ugyanazt a címet adja, ezeket a különbözõ kulcsú rekordokat szinonimoknak mondjuk. A szinonim rekordokat kezelni kell (hogy hogyan azzal most nem foglalkozunk). Nyilván annál jobb egy hash leképezés minél kevesebb szinonim van.

Példa hash leképezésre: alkalmasan nagy prímszámmal osztjuk a numerikusnak tekintett kulcsot, és a maradék lesz a cím.

Kulcs szerinti közvetlen feldolgozásra a jó hash leképezés hatékonyabb az indexnél, ezért a hash leképezést elõszeretettel alkalmazták a hierarchikus és hálós modellben.

A relációs adatbázisoknál nem használják, mert:

- nem tesz lehetõvé szekvenciális elérést
- csak kulcs szerint lehet keresni, egy rekord más mezõje szerint nem.

6. Az adatbázis-tervezés fázisai

Az adatbázis-tervezés fázisai a következõk:

1. A feladat specifikációja
2. Koncepcionális terv
3. Adatbázis-kezelõ rendszer kiválasztása
4. A 2. fázisban elkészült magas szintû modell leképezése (hálós, hierarchikus vagy relációs modellre)
5. Fizikai terv
6. Megvalósítás

Napjainkban a relációs adatbázis-kezelõk népszerûsége kapcsán rendszerint a feladat elkezdésekor adott, hogy a feladatot relációs adatmodellben kell megtervezni, ilyenkor az adatbázis megtervezésének fázisai:

Az adatbázis-tervezés fázisai már a feladat elején a relációs adatmodellt feltételezve	
I. verzió	**II. verzió**
1. A feladat specifikációja 2. Koncepcionális terv 3. A 2. fázisban elkészült magas szintû modell leképezése 4. Konszolidáció 5. Normalizálás, ha szükséges 6. Fizikai terv 7. Megvalósítás	1. A feladat specifikációja 2. Normalizálás 3. Konszolidáció 4. Fizikai terv 5. Megvalósítás

6.1 A feladat specifikációja

A tárolandó adatok és mûveleti igények összegyûjtése

- a jelenlegi megoldások megismerésével
- interjúkkal az egyes felhasználók igényeit
- az adott területtel rokon megoldások tanulmányozása

Ebben a fázisban az ún. CASE eszközök segíthetik a munkát.

6.2 Koncepcionális terv

Adatbázis-kezelõtõl független koncepcionális (pl. ER) modell kialakítása kétféle stratégiával:

1. globális stratégiával
2. nézet integráló stratégiával.

6.2.1 Globális stratégia

1. Az egyes felhasználói csoportok igényeinek összegyûjtése.
2. Az igények összefésülése.
3. Koncepcionális (pl. ER) modell elkészítése a teljes rendszerre.

6.2.2 Nézet integráló stratégia

1. A csoportok igényeinek (nézetigények) összegyûjtése.
2. Minden csoportigényre koncepcionális modellt (alsémát) építünk.
3. Az alsémákat összefésüljük, hogy felrajzoljuk a teljes rendszer koncepcionális modelljét.
4. Elvégezzük a szükséges szerkezeti változtatásokat.

6.3 Adatbázis-kezelõ rendszer kiválasztása

Az adatbázis-kezelõ rendszer kiválasztásában leginkább két tényezõ játszik szerepet:

1. a feladat természete (pl. ha sok N:M kapcsolat van, akkor a hierarchikus adatbázis-kezelõ rendszer nem alkalmas)
2. gazdaságossági szempontok.

6.4 Magas szintû modell leképezése

Az egyes adatbázis-kezelõtõl függõ adatmodellek (hálós, hierarchikus, relációs) tárgyalásánál szót ejtettünk azokról a leképezési szabályokról, amelyek a magas szintû adatmodellt (pl. ER modell) képezik le alacsonyabb szintû (hálós, hierarchikus, relációs) modellre.

A leképezés automatizálható az ún. CASE eszközök segítségével.

A leképezés után kapott relációs modell általában harmadik normálformában van, de esetleg a normalizálás további alkalmazásával tovább finomíthatjuk a kapott adatmodellt.

6.5 Fizikai terv

Ebben a fázisban kell megadni az adatok fizikai tárolási és elérési módját.

Célszerû a rekordokat a rendezési attribútum (általában az elsõdleges kulcs) szerint fizikailag is rendezni.

A relációs modellben a fizikai tervezés fontos feladata az indexelés:

A lekérdezések meggyorsítása miatt érdemes az egyes relációkat indexelni:

- a feltételekben szereplõ mezõk szerint
- összekapcsoló mezõk szerint.

Nem célszerû indexelni azon mezõk szerint, amelyeknek gyakran változik az értéke. Túl sok mezõ szerint nem érdemes indexelni a relációkat, mert az indexfájlok mérete miatt túlzottan megnõhet az adatbázis mérete.

6.6 Megvalósítás

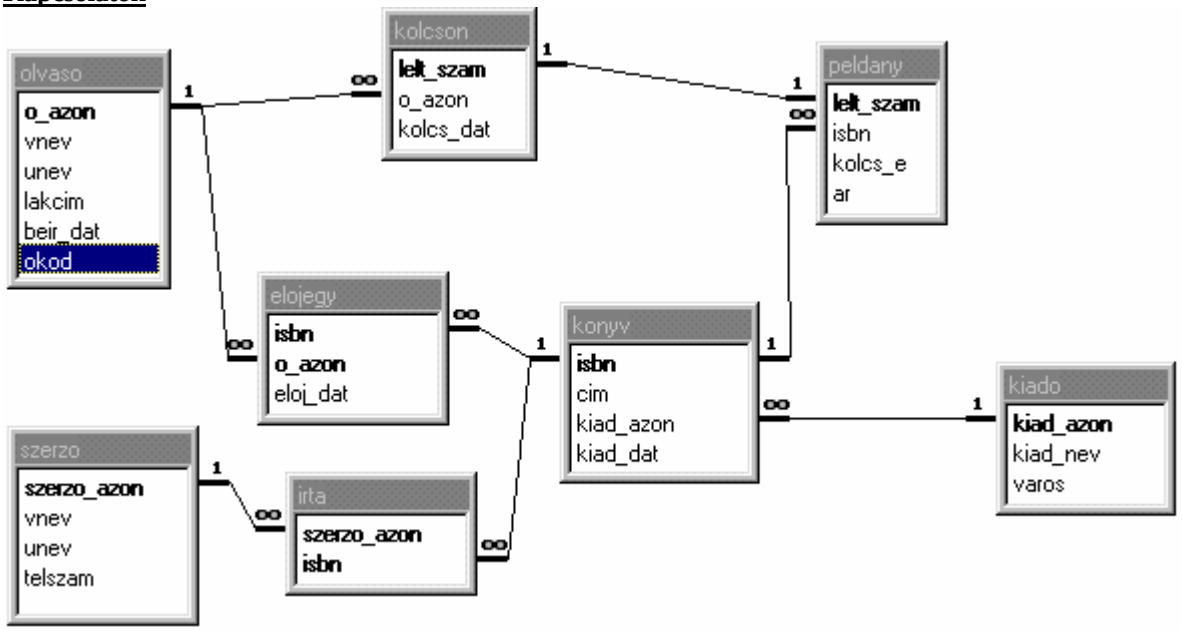
SQL nyelvvel:

- DDL nyelv segítségével leírjuk az adatbázis szerkezetét.
- DCL nyelv alkalmas arra, hogy gondoskodjunk az adatvédelemrõl.
- DML nyelv segítségével feltöltjük az adatbázist.
- QL nyelv segítségével lekérdezések végezhetõk

Függelék

Függelék A Függelék B Suli-könyvtár adatbázis

Kapcsolatok



A táblák adatai

OLVASO

O_AZON	VNEV	UNEV	LAKCIM	BEIR_DAT	OKOD
001	GIPSZ	JAKAB	DEBRECEN FAL U. 1.	04-JAN-90	
002	KEMENY	HELEN	APAFA FA U. 12.	27-FEB-95	
003	MINTA	MOKUS	SARAND FELFAL U. 9.	30-NOV-94	7
004	KEREK	ERNO	SZOB TINTA U.13.	22-MAY-93	6
005	POR	OSZKAR	EGER DOBO U.21.	12-MAY-93	6

KIADO

KIAD_AZON	KIAD_NEV	VAROS
K001	TANKONYVKIADO	LONDON
K002	AKADEMIAI KIADO	NEW YORK
K003	GONDOLAT KIADO	LONDON
K004	KOSSUTH KIADO	LONDON

KONYV

ISBN	CIM	KIAD_AZON	KIAD_DAT
100001	TUSKEVAR	K002	01-JAN-93
100002	EGRI CSILLAGOK	K001	12-FEB-97
100003	KOSZIVU EMBER FIAI	K001	21-JUN-94
100004	EMPATIA	K003	24-NOV-91
100005	ANATOMIA	K002	01-JUL-90
100006	RECEPTEK	K003	01-JUL-92

PELDANY

LELT_SZAM	ISBN	KOLCS_E	AR

L001	100001	1	1100	
L002	100001	1	1100	
L003	100001	1	1150	
L004	100002	1	800	
L005	100002	1	800	
L006	100003	0	1200	
L007	100004	1	300	
L008	100005	1	650	
L009	100004	0	300	
L010	100004	1	340	
L011	100005	0	680	
L012	100006	1	600	
L013	100006	1	600	

KOLCSON

LELT_SZAM	O_AZON	KOLCS_DAT
L002	002	05-JAN-97
L003	003	28-JUL-97
L004	002	21-JUN-97
L005	001	11-AUG-97
L007	002	15-AUG-97
L008	001	21-FEB-97

ELOJEGY

ISBN	O_AZON	ELOJ_DAT
100002	003	22-AUG-97
100005	002	21-JUN-97

SZERZO

SZERZO_AZON	VNEV	UNEV	TELSZAM
S01	FEKETE	ISTVAN	
S02	GARDONYI	GEZA	
S03	JOKAI	MOR	
S04	BUDA	BELA	
S05	TARSOLY	EMIL	
S06	KUDLIK	JULIA	
S07	PSOTA	IREN	

IRTA

SZERZO_AZON	ISBN
S01	100001
S02	100002
S03	100003
S04	100004
S04	100005
S05	100005
S06	100006
S07	100006

DOLGOZO

d_azon	Vnev	Unev	beosztas	belepes	fizetes	fonok
D01	NAGY	KLARA	IGAZGATO	30-NOV-92	110000	NULL
D02	KISS	TEREZ	OSZTALYVEZETO	13-JAN-94	82000	D01
D03	BARNA	PETER	OSZTALYVEZETO	23-SEP-93	79000	D01
D04	SZILARD	ISTVAN	KONYVTAROS	17-MAR-91	28000	D02
D05	KEREK	EMIL	KONYVTAROS	10-OCT-92	31000	D03
D06	FUTO	ERZSEBET	ELJARO	01-FEB-96	30000	D01

Függelék C Tematika

I2924 Adatbázis-kezelés

Tematika: Az adatbázis kezelõ rendszerek kialakulása. Egy általános adatbázis rendszer architektúrája. adatmodellezési stratégiák. Relációs és CODASYL adatmodell. Adatdefiníciós és adatmanipulációs nyelvek. Önálló és befogadó nyelvû rendszerek. Az SQL. A tárgy gyakorlatán ismertetésre kerül egy konkrét könyvtári információs rendszer.

Függelék D Rövidítések

ABR	Adatbázisrendszer
DDL	Data Definition Language (adatdefiníciós nyelv)
DML	Data Manipulation Language (adatmanipulációs nyelv)
DCL	Data Control Language (adatvezérlõ nyelv)
SQL	Structured Query Language (struktúrált lekérdezõ nyelv)
DB	Data Base (adatbázis)
DBA	Data Base Administrator (adatbázis adminisztrátor)
DBMS	Data Base Management System (adatbázis-kezelõ rendszer)
RDBMS	Relational Data Base Management System (relációs adatbázis-kezelõ rendszer)
CASE	Computer Aided Software Engineering
DBS	Data Base System
ER	Entity Relationship
PCR	Parent Child Relationship
VPCR	Virtual Parent Child Relationship
QL	Query Language
SQL	Structured Query Language
CDD	Common Data Dictionary