A relációs kalkulus 1.

Létezik sor- és oszlopkalkulus, mi csak az előbbivel foglalkozunk.
A kalkulus az alábbi alakú kifejezésekből áll:

$$\{t \mid \psi(t)\}$$

melynek jelentése: azok a t sorok, melyek kielégítik a ψ(t) függvényt, azaz amelyekre a ψ(t) függvény értéke igaz.

A relációs kalkulus 2.

Atomi formulának, vagy atomnak nevezzük, az alábbi kifejezéseket:

1.) R(s) **(azaz az s sor R relációbeli),**

2.) s[i] Θ u[j] **(ahol s[i] az s-edik sor i-edik elemét, u[j] az u-adik sor j-edik elemét jelenti, Θ a szokásos összehasonlító operátor),**

3.) s[i] Θ c **(ahol c egy konstans).**

A relációs kalkulus 3.

A ψ(t) függvény, melyet *formulának* nevezünk, az alábbiak szerint épülhet fel:

1.) Minden atom formula.

2.) A formulákból a ∧, ∨ , ¬ logikai műveletekkel képezett kifejezések is formulák.

3.) A (∃s)(ψ) alakú (van ilyan s, hogy ψ igaz) kifejezések is formulák.

A relációs kalkulus 4.

4.) A $(\forall s)(\psi)$ alakú (minden s olyan, hogy ψ igaz) kifejezések is formulák.

5.) A kifejezések a precedencia (Θ, $\exists$, $\forall$, $\neg$, $\wedge$, $\vee$) megváltoztatása érdekében zárójelezhetők.

6.) Más formula nincs.

A relációs algebra és kalkulus összehasonlítása 1.

A relációs algebra alapműveletei kifejezhetők a relációs kalkulus eszközeivel.

Unió: $\{t \mid R(t) \vee S(t)\}$

Különbség: $\{t \mid R(t) \wedge \neg S(t)\}$

Direkt szorzat: $\{t^{(r+s)} \mid (\exists u^{(r)})\ (\exists v^{(s)})(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge \ldots \wedge t[r]=u[r] \wedge t[r+1]=v[1] \wedge \ldots \wedge t[r+s]=v[s]\ \}$

Projekció: $\{t^{(k)} \mid (\exists u)(R(u) \wedge t[1]=u[i_1] \wedge \ldots \wedge t[k]=u[i_k]\ \}$

Szelekció: $\{t \mid (R(t) \wedge F\ \}$

A relációs algebra és kalkulus összehasonlítása 2.

$\{t \mid \neg R(t)\}$
értelmetlen, de nem írható le a rel. algebrában.

DOM fv.:
A DOM(ψ) azon elemek halmaza, melyek közvetlen, vagy közvetett módon ψ-t alkotják.

A relációs algebra és kalkulus összehasonlítása 3.

Biztos kifejezések:

1. **Ha t kielégíti ψ(t)-t, t minden komponense DOM(ψ)-beli.**
2. **ψ minden (∃u)(ω(u)) alakú részkifejezésére, ha u kielégíti ω(u)-tu minden komponense DOM(ω)-beli.**

A rel. kalkulus biztos kifejezései ekvivalensek a rel. algebra kifejezéseivel.

Lekérdezés relációs rendszerekben 1. (ISBL)

Korai IBM termék.

Lényegében a relációs algebra kifejezéseinek linearizálása.

Nem felhasználóbarát.

Lekérdezés relációs rendszerekben 2. (QBE)

IBM termék.

Tábla vázakat (skeleton) használ, ezeket vektorváltozók segítségével lehet összekapcsolni.

Példa QBE-re 1.

AUTÓK

rendsz.	gym.	szín	. . .
	OPEL	SÁRGA	

Példa QBE-re 2.

AUTÓK

rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

Példa QBE-re 3.

AUTÓK

rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

KAPCSOLAT

frsz.	sz.sz.	. . .
X	Y	

Példa QBE-re 4.

AUTÓK

rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

KAPCSOLAT

frsz.	sz.sz.	. . .
X	Y	

EMBEREK

sz.sz.	név	fogl.	. . .
Y	Kiss	tanár	
	…	…	

Lekérdezés relációs rendszerekben 3. (SQL)

Structured Query Language

Több részből áll:

1. DDL

2. DCL

3. DML

4. Query

DDL

CREATE TABLE
ALTER TABLE
DROP TABLE

CREATE VIEW
DROP VIEW

CREATE [UNIQUE] INDEX
DROP INDEX

DCL

Tranzakció kezelés:
COMMIT
ROLLBACK
LOCK
UNLOCK

Jogosítvány kezelés:
GRANT …. WITH GRANT OPTION
REVOKE

DML

INSERT
UPDATE
DELETE

QUERY

A SELECT utasítás valósítja meg.
Legegyszerűbb formája:

select [distinct] ***oszlopnevek*** **from** ***táblanevek***

[where ***feltétel*****]**

[order by ***rendezési feltétel*****]**

[group by ***feltétel*** **[having** ***having- feltétel*****]]**

Keresés egymásba ágyazott SELECT-ek esetén

A WHERE részben alkalmazható újabb SELECT klauzula:

1. [NOT] IN (selectkifejezés)

2. Θ[ANY|ALL] (selectkifejezés)

3. [NOT] EXISTS

Több SELECT esetén a kiszámítás belülről kifelé történik.

Beágyazott (embedded) SQL

1. EXEC SQL <beágyazott SQL utasítás>

2. Változók használata V→:V

3. Vezérlés:

$$\text{WHENEVER} \left\{ \begin{matrix} \text{NOT FOUND} \\ \text{SQL WARNING} \\ \text{SQL ERROR} \end{matrix} \right\} \left\{ \begin{matrix} \text{GO TO cimke} \\ \text{CONTINUE} \\ \text{STOP} \end{matrix} \right\}$$

4GL program generátorok

1. **FORM (űrlap)**
2. **REPORT (jelentés)**
3. **MENU**

Továbbfejlesztett modellek

1. **Az EER modell.**
2. **Nested Relational Model.**
3. **Structural Data Model.**
4. **Objektum-orientált adatbázisok.**
5. **Deduktív adatbázisok.**

Az EER modell

Subclass, superclass bevezetése.
(Közös tulajdonságok, kapcsolatok leírása.)

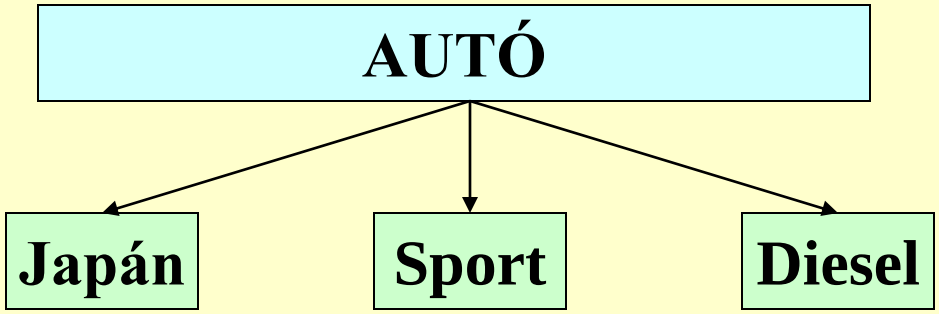
A hierarchiában a superclass tulajdonságai öröklődnek (inheritance).

Relációkkal megvalósítható.

Az EER modell felosztása

1. A subclass lehet attribútum által meghatározott, vagy felhasználó által definiált.
2. A felosztás lehet diszjunkt (disjoint), vagy átfedő (overlap).
3.) Ugyancsak lehet a felosztás teljes (complett) vagy részleges (partial).

Példa EER modellre



Superclass

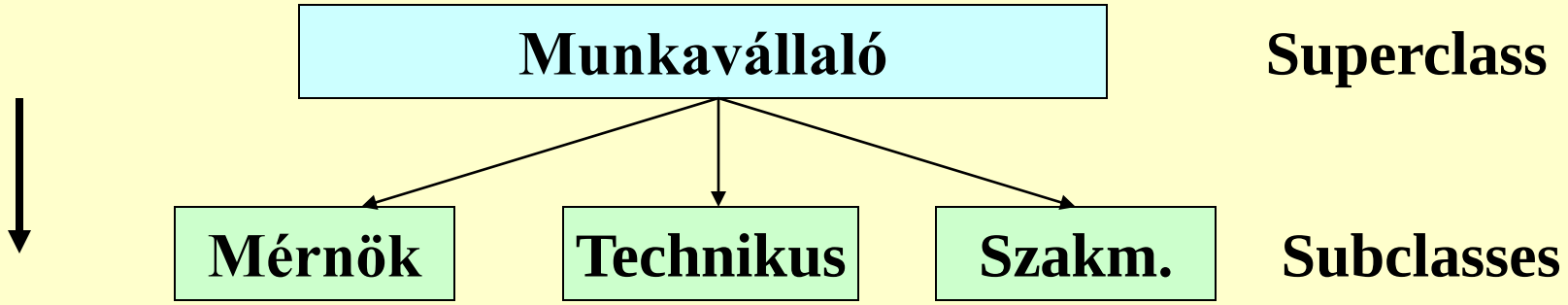
Subclasses

A subclass-ok nem diszjunktak,
a felosztás részleges!

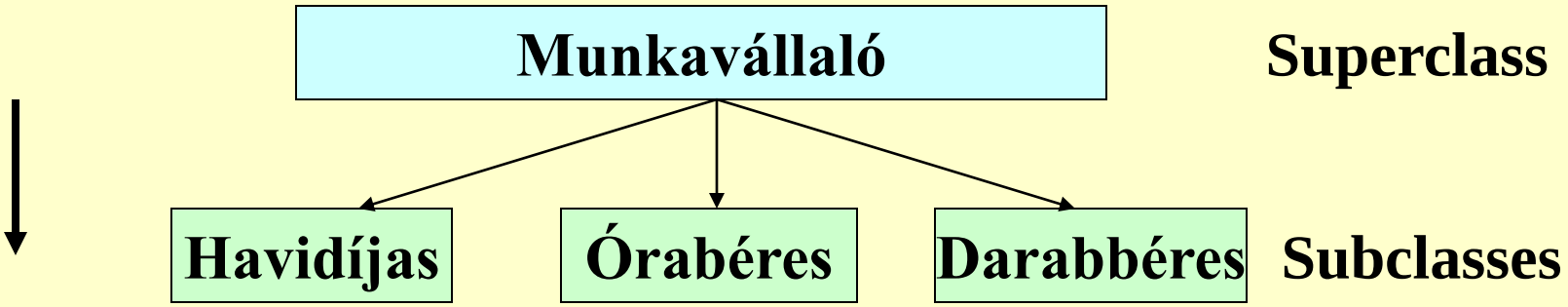
Az EER modell használata

1.) Specializáció
Megkeressük az egy-egy csoportra jellemző tulajdonságokat.
2.) Generalizáció.
Megkeressük a közös tulajdonságokat.

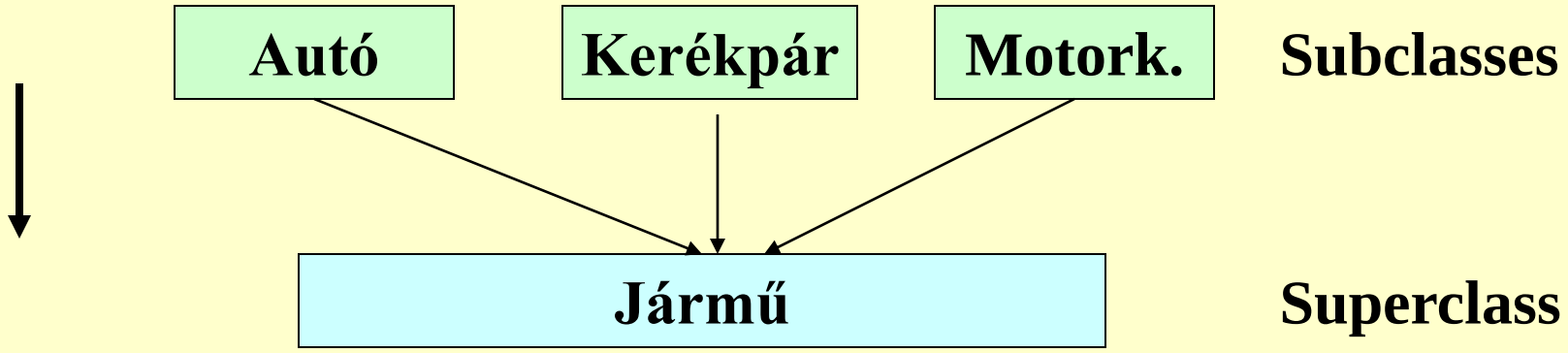
Példa Specializációra



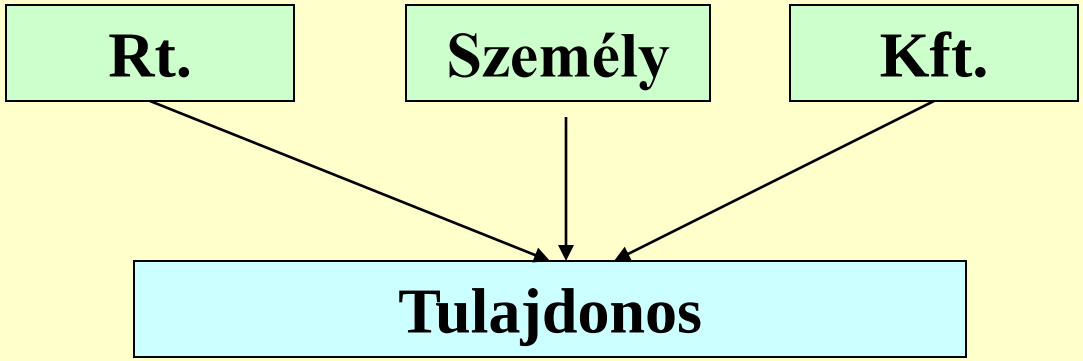
Lehet több szempont szerint is:



Példa Generalizációra



Kategória



Kategória az a subclass, melynek több superclass-a van.
Itt az öröklés szelektív lehet.

Nested Relational Model

Nem 1NF, ezért nevezik N1NF modellnek is.

Egy osztály sémája

<table>
<tr><td rowspan="3">Oszám</td><td rowspan="3">Onév</td><td rowspan="3">Ovez</td><td colspan="3">Dolgozó</td><td rowspan="3">Ocím</td></tr>
<tr><td rowspan="2">Dnév</td><td colspan="2">Gyerekek</td></tr>
<tr><td>Gynév</td><td>Gykor</td></tr>
</table>

Structural Data Model 1.

Az eredeti relációs modell továbbfejlesztése, az SQL2 alapja.

Két típus:
1.) Relations
2.) Connections

Structural Data Model 2.

Relations

1.) Primary: nem kapcsolódik hozzá semmi.

2.) Referenced: kapcsolódik hozzá reláció.

3.) Nest: többértékű attribútumot tartalmaz.

4.) Association: M:N kapcsolatot reprezentál.

5.) Lexicon: egy az egyhez kapcsolat az attribútumok között.

6.) Subrelation: csak egy másikkal együtt van értelme.

Structural Data Model 3.

Connections

1.) Ownership: tulajdonos és a hozzátartozó nest, vagy association reláció között.

2.) Reference: referenced relations-ok között.

3.) Identity: reláció és subrelation között, azaz ahol a primary key és a foreign key azonos.

OODB 1.

Az ötlet az OOP alapján keletkezett.

Jellegzetességek:

1.) Az objektumosztályok perzisztensek.

2.) Az objektumosztályok osztottak.

3.) Minden objektumnak külön azonosítója (OID) van (nem azonos a kulccsal, az különbözhet relációnként).

4.) Megengedettek a bonyolult objektumok (pl. egy objektumon belüli több reláció, stb.).

OODB 2.

Encapsulation

Az értékeket u.n. *instance variable*-ok tartalmazzák. ez hasonlít az attributum –hoz, de kívülről rendszerint nem látható, csak az előre definiált *operációk* férhetnek hozzá (metódus).

Egy operációnak két része van:

***1.)Signature*, vagy *interface*: a név és a paraméterek.**

2.)*Method*, vagy *body*: az implementáció.

OODB 3.

Inheritance, polymorphism

A szokásos. Érdekes az operátorok polimorfizmusa, azaz az a tulajdonság, hogy egy operátor névhez többféle implementáció tartozhat az objektum típu-sától függően (másnéven *operator overloading*. Kell hozzá a late binding!).

Kapcsolatok

Az encapsulation miatt nehézségek adódhatnak.

1. **Kapcsolatot kereső metódusok.**
2. **References:az objektum tartalmazza a kapcsolt objektumok OID-jét.**

Az O2 adatdefiniálás I.

A séma objektum típusokat és osztályokat definiál.
Az objektum típusokat az u.n. atomi típusok használatával és az O2 típus konstruktorok alkalmazásával definiálhatjuk.
Atomi típusok: Boolean, character, integer,real, string és bits.
A típus konstruktorok: tuple, list, set, unique set.
Az O2 metódusok nem részei a típus definíciónak.
Az osztály definíció két részből áll:típus és metódus megadásból.

O2 adatdefiniálás II.

A Különbség van az O2-ben értékek (value) és objektumok között.

Az értéknek csak típusa van, és önmagát reprezentálja.

Az objektum egy osztályhoz tartozik és így van egy típusa és vannak metódusai, melyeket az osztály határoz meg.

Mind az értékek, mind az objektumok komplex típusúak, és ezek lehetnek értékek, vagy lehetnek hivatkozások más objektumokra az OID használatával.

O2 adatdefiniálás III.

Az O2-nek van egy saját nyelve az O2C, ezen lehet definiálni osztályokat, metódusokat, típusokat, továbbá létrehozni (create) objektumokat és értékeket.

Az objektumok lehetnek perzisztensek (állandóan tárolva vannak az adatbázisban) és tranziensek (csak egy program végrehajtása alatt léteznek), az értékek tranziensek, (hacsak nem részei egy perzisztens objektumnak).

O2 adatdefiniálás példa

```
type telefon: tuple     (körzetszám: integer,
                         telefonszám: integer);

class személy:
        type tuple(      név: tuple (vezetéknév:string,
                                     keresztnév:string),
                         szülinap: Date,
...
        method           kor:integer
end
```

Az O2 adatmanipulálás

Adatmanipulálási lehetőségek:

- **az O2SQL lekérdező- és az O2C program nyelv,**
- **beágyazott pl. C++-ba.**

Fejlesztő környezetek:

- **O2Look (interface O2C-hez),**
- **O2Tools (grafikus fejlesztői környezet).**

Az Objectstore rendszer

A C++ nyelvhez készült, annak objektum deklarációs utasításait használja.

Adatmanipuláláshoz is a C++-t használhatjuk, de van grafikus interface is.

Deduktív adatbázisok

A logikai programozás eszközeivel dolgozik. Deklaratív nyelvet (pl. PROLOG) használ. ezen írja le a *tényeket* és a *szabályok*at.

Adatbázis kezelő architektúrák

Három fő rész:

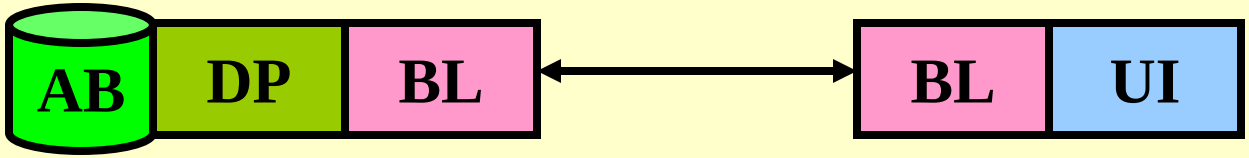
1.) Data processing (fizikai adatkezelés).

2.) Business logic (adatvédelem és –integritás, tranzakció kezelés, stb.).

3.) User interface

Hol helyezkednek el?

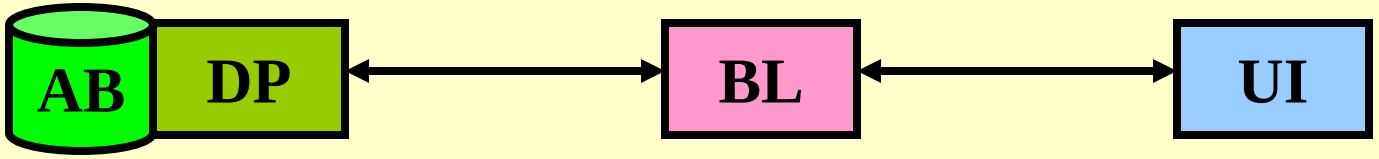
Kliens-szerver architektúra



Adatbázis szerver

Kliens alkalmazás

Többrétegű architektúra



Adatbázis szerver

Középső réteg (middle-tier)

„Sovány” („Thin”) kliens alkalmazás

Osztott adatbázisok

Megnövekedett a kommunikációs költségek részaránya. Javaslat: helyezzük az adatokat a felhasználóhoz közel.

Osztott (distributed) adatbázis:

fizikailag megosztott, de logikailag egységes adatbázis.

Adatbázis felügyelet: központi, csomóponti.

Osztott adatbázisok előnyei

1.) A kommunikációs költségek csökkenése.
2.) Mindenki a számára ismerős adatokat gondozza.
3.) Egy-egy csomópont kiesése esetén a többi adatai továbbra is elérhetőek.
4.) Lehetséges a moduláris tervezés, a rugalmas konfigurálás.
5.) Hosszabb idő alatt a rendszer gépei akár ki is cserélhetők.

Osztott adatbázisok hátrányai

1.) A rendszer bonyolultabb és sebezhetőbb lesz,
2.) Nem könnyű minden csomópontra egyformán jó személyzetet találni, másrészt, ha találunk fenyeget a szuboptimalizáció veszélye.
3.) Mindig valamennyi gépnek működnie kell.
4.) Többféle hardvert és szoftvert kell a rendszernek kezelnie és "összehoznia".
5.) Bonyolult a jogosultságok ellenőrzése.

Osztott adatbázisok konzisztenciája

Külön problémát jelent, ha feladjuk a redundancia-mentesség elvét. Erre okot szolgáltathat az is, ha nem eldönthető egy-egy adatról, hogy hol használják legtöbbet, de biztonsági okokból is dönthetünk egy-egy adat többszörözése mellett. Ilyen estekben biztosítanunk kell, hogy az egyes példányok tartalma azonos legyen, azaz a rendszer *konzisztens* maradjon.

Elemzések

1. Forrás nyelő elemzés

   A használat gyakorisága, módja.

2. ABC elemzés

   A „nélkülözhetetlenség” foka.

3. Érzékenység elemzés

   A költség/teljesítmény arány.

Konzisztencia, konvergencia

Létezik

- **Erős konzisztencia**
- **Gyenge konzisztencia**

Koherencia: a konzisztencia mérő száma, mely erős konzisztenciánál azonosan 1 értékű, gyenge konzisztenciánál pedig 1-hez tart.

Szinkronizációs protokollok

A. Központosított protokollok

a.) A központi zárellenőrzés

b.) A zseton módszer

c.) Az elsődleges példány módszer

B.) Osztott protokollok

Az időbélyeg módszer

Adatvédelem

a.) Fizikai védelem

b.) Ügyviteli védelem

c.) Algoritmikus védelem

Algoritmikus védelem

a.) felhasználó azonosítás, partner azonosítás

b.) hozzáférés védelem

c.) rejtjelezés

d.) üzenethitelesítés

e.) digitális kézjegy

Rejtjelezés

$y = E(x)$ → Védetlen közeg → $x = D(y)$

Legyen „megfejthetetlen”.

Mivel sok kell, legyen:

$y = E(k_r,x)$ → Védetlen közeg → $x = D(k_f, y)$

A rejtjelezés felosztása

1. **Konvencionális a rejtjelezés, ha a rejtő kulcsból a fejtő kulcs meghatározható.**

2. **Nyílt (nyilvános) kulcsú a rejtjelezés, ha rejtő kulcsból a fejtő kulcs nem határozható meg.**

A konvencionális kódolás

a.) Helyettesítés

b.) Periodikus helyettesítés

c.) Kulcsfolyam(at)os rejtés

d.) Rejtjelötvözés vagy keverő transzformációk

(Shannon, 1949)

Lucifer (128 bites blokk, 128 bites kulcs)

DES (64 bites blokk, 56 bites kulcs)

A nyilvános (rejtő) kulcsú kódolás

a.) MIT módszer (prímfelbontás).

b.) Merkle-Hellmann módszer (hátizsák problé-ma).

Kulcsgondozás

Ha a kulcsok megfelelő védelme nem biztosított, az egész kódolási rendszer értelmetlen. A kulcsok gondozása három feladatból áll.

a.) kulcsgenerálás

b.) kulcskiosztás

c.) kulcstárolás

Kulcsgenerálás

Az a művelet, melynek eredményeképpen a kulcsok előállnak. Egy valódi véletlenszám generátor segítségével végrehajtható.

Kulcskiosztás

Példák:

a.) Alapkulcsok

b.) Merkle „rejtvény” módszere

c.) A "hatványozós" módszer

Alapkulcsok

A kulcsok kiosztása történhet egy *alapkulcs készleten* keresztül, melyet rendszeren kívüli eszközökkel juttatnak el a résztvevőkhöz. Az alapkulcsokat, melyek gyakran nyilvános kulcsú rendszerhez tartoznak, csak a kulcsok cseréjéhez használják.

Merkle „rejtvény” módszere

A hívó fél n db (K_i, I_i) párt küld partnerének gyengén kódolva. Ez egyet kiválaszt, feltöri, és I_i-t visszaküldi. Ezzel a kommunikáció kulcsa meghatározott. A behatolónak átlagosan a párok felét kell ahhoz feltörnie, hogy megtudja, I_i-hez melyik K_i tartozik.

A "hatványozós" módszer 1.

A módszer alapja, hogy az i és a j felhasználó kitalál egy-egy x_i ill. x_j számot.

Egymás között kicserélik az

$$Y_i = \alpha^{x_i} \pmod p,$$

ill. az

$$Y_j = \alpha^{x_j} \pmod p$$

számokat (p prímszám, a p elemű véges test egy primitív eleme).

A "hatványozós" módszer 2.

A a kommunikáció kulcsa a

$$K = \alpha^{x_i * x_j}$$

szám (vagy annak valamilyen függvénye) lehet. Ennek előállítása α mellett Y_j és x_i vagy Y_i és x_j ismeretében egy egyszerű hatványozást igényel, a behatolónak viszont a diszkrét logaritmusképzés a feladata.

Kulcstárolás

Nehézségeket okozhat, ha a kulcsokat akár túl kevés, akár túl sok ember ismeri.

Megoldást jelentenek az u.n. *(n,k) küszöbrendszerek*. E rendszerek lényege, hogy a kulcsot n db. (nem feltétlenül diszjunkt) részre osztva, bármelyik k kulcsrészletből a kulcs előállítható, de nincs olyan k-1 kulcs-részlet, amiből ez megtehető lenne. Ilyen küszöbrendszerek készítésére több matematikai módszer is rendelkezésünkre áll.

Felhasználó azonosítás

Személy azonosítás:

a.) jelszóvédelem

b.) fizikai azonosító használata

c.) személyi jellemzők

Partner azonosítás 1.

A számítógép-számítógép kapcsolatokban is szükség lehet azonosításra. E célra fenntarthat minden számítógép pár egy-egy kulcsot, ez azonban egy n elemű hálózatnál n^2 kulcsot jelent. Folyhatnak az információcserék valamilyen hitelesítő központon keresztül, ez azonban a kommunikáció költségét növeli jelentősen.

Partner azonosítás 2.

Megoldás:ha minden csomópont egy, csak a hitelesítő központ által ismert kulccsal tud e központhoz bejelentkezni, s üzenet továbbítási igényét bejelente-ni. A központ jelöli ki a kommunikáció kulcsát, melyet a fenti kulccsal kódolva a hívónak visszaküld. Emellett küld egy csomagot, mely a hívandó fél kulcsával kódolva tartalmazza a kommunikáció kulcsát, s a hívó megjelölését. Ha a hívó e csomagot a hívott félhez továbbítja a párbeszéd kettejük között folytatódhat, s az azonosítás is kielégítő biztonsággal történt meg.

Hozzáférésvédelem

Nem elegendő kiszűrni a jogosulatlan behatolókat, a rendszernek azt is számon kell tartania, hogy a jogos felhasználók hatásköre mire terjed ki. A felhasználó által működtetett *ügynök folyamatok* hatáskörét a *hozzáférési mátrix* szabja meg. Ennek elemeit akár ügynökökhöz, akár adatokhoz kötötten tárolhatjuk.

Üzenethitelesítés

Az üzenetek hitelesítéséhez két feltétele van, egyrészt ellenőrizni kell, hogy egy-egy blokkban az és csak az érkezett-e a címzetthez amit a feladó feladott, másrészt tudnunk kell azt is, hogy az egyes blokkok abban a sorrendben érkeztek e meg amilyenben feladták őket (ideértve azt is, hogy nem hiányzik-e közülük). Az első célhoz ellenőrző összegeket, a másodikhoz sorszámot célszerű a blokkban elhelyezni még a kódolás előtt.

A digitális kézjegy

arra szolgál, hogy segítségével a címzett megbizonyosodhassék egy üzenet feladójáról és bizonyíthassa, hogy az illetőtől kapott ilyen üzenetet. Olyasmire van szükség tehát mint az aláírás, ami könnyen azonosítható, de nehezen hamisítható.

A cél elérhető úgy is, hogy a kényes kommunikációt egy hitelesítő központ közbeiktatásával végezzük (mintha tanú előtt beszélnénk) ez az u.n. *nem valódi digitális kézkegy*.

A valódi digitális kézjegy

Ehhez egy nyilvános kulcsú kódolási rendszer lehet felhasználni, mégpedig olyant, mely "megfordítható", azaz E(D(x)) = x (az általunk említett módszerek ilyenek). Rejtsünk a titkos fejtő kulccsal (és fejtsünk a nyílt rejtő kulccsal). A címzett, ha a nyílt kulccsal fejthető üzenetet kap, biztos lehet abban, hogy azt csak a titkos kulcsot ismerő feladó küldhette. Mivel a címzett nem ismeri a titkos kulcsot, ha rendelkezik az üzenetnek a nyílt kulccsal megfejthető kódolt változatával, nyilvánvaló, hogy azt ő nem készíthette.

Hagyományos igények

OLTP (On Line Transaction Processing):

- az ábrázolt mini világ minden adatát tartalmazza
- az utolsó állapotot mutatja
- sok adatmódosítás
- egy-egy tranzakció kevés adatot érint
- viszonylag egyszerű, de ad hoc kérdésekre is tud válaszolni
- a válaszidő kicsi
- jellemző több, párhuzamosan működő felhasználó

Új igények

1. **Adatfolyamok feldolgozása.**

2. **Előre elkészített, aggregált adatok a vezetőknek:**
   DSS (Decision Support System).

3. **Nem ismert összefüggések kiderítése:**
   Tudásfeltárás (Data Mining, adatbányászat)

Adatfolyamok feldolgozása I.

Pl. szenzorok állandóan tömegével generál adatokat, ezeket nem tároljuk mind le, de kérdéseink lehetnek.

Forgalom figyelés: hálózati, banki, közleke stb.

Naplózás: meteorológiai, egészségügyi, ipar

Adatfolyamok feldolgozása II.

Új eszközök (részben az SQL kiterjesztésé

Stanford University:
CQL (Continuus Query Language)

Cornell University:
COUGAR

Adatfolyamok feldolgozása III.

1. Az ilyen kérdések hosszú ideig működne
2. Sokszor a megválaszoláshoz hagyomány adatbázis is kell.

Probléma: hogyan kezeljük az adatbázis adatainak változását?

(Pl. kereskedelmio forgalom vizsgálata, de árak és az ügyfelek címei hagyományos adatbázisban.)

DSS (Decision Support System)

OLAP (On Line Analitical Processing)

Megoldásai:

ROLAP
MOLAP

OLAP

- nem feltétlenül egészen up-to-date
- csak az elemzéshez szükséges adatokat tartalmazza, ezek azonban több mini világból származnak
- tartalmazza a régi adatokat (trendek)
- jellemzően olvas, de bonyolult elemzéseket végez
- a válaszidő nem kritikus
- látványos riportok, ezek könnyen elérhetőek (intra- internet, wap, sms, stb.)

ROLAP

Relational On Line Analitical Processing:

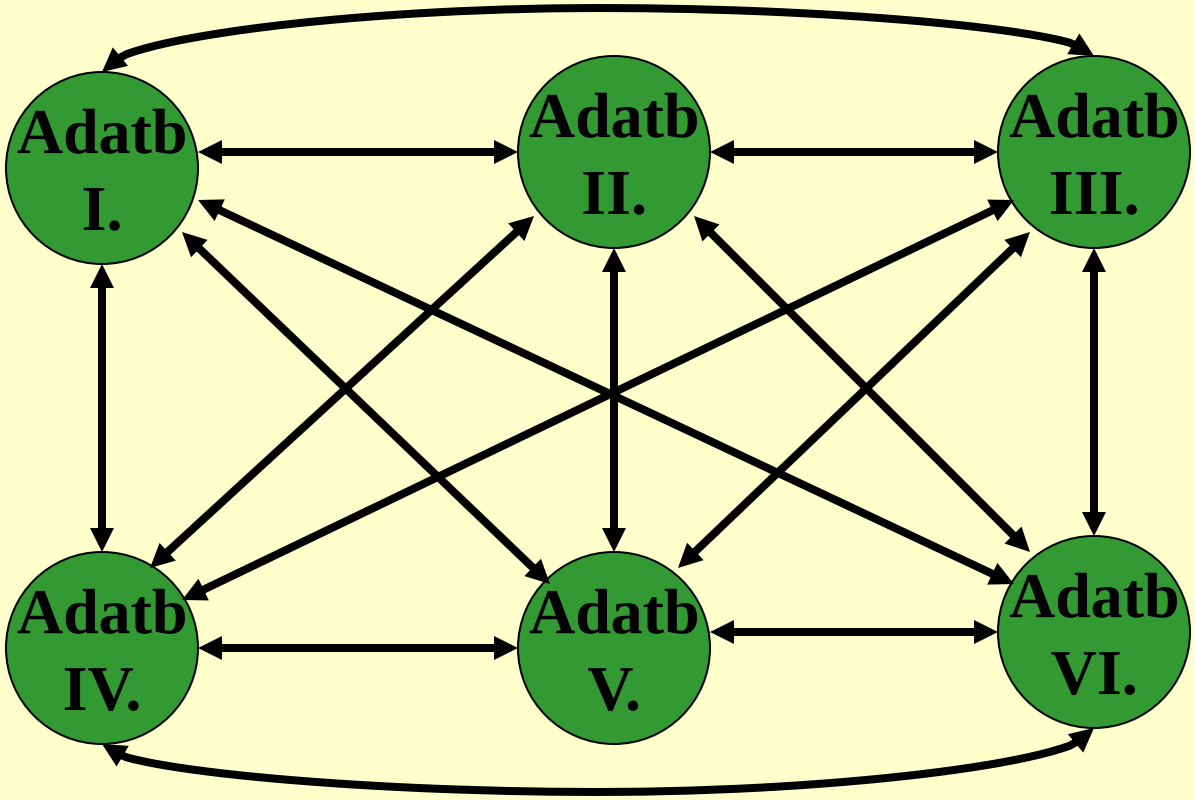
a jól ismert és bevált relációs eszközöket használja, ezek azonban nem erre a célra készültek.

MOLAP

Multidimensional On Line Analitical Processing:

Az adatokat egy többdimenziós kockában tárolja.
Könnyű megvizsgálni egy kiválasztott élnek a többitől való függését.
- lassan kiépíthető, hardware igényes rendszer
- gyorsan ad választ a várt kérdésekre.

Adatbázisok összekapcsolása

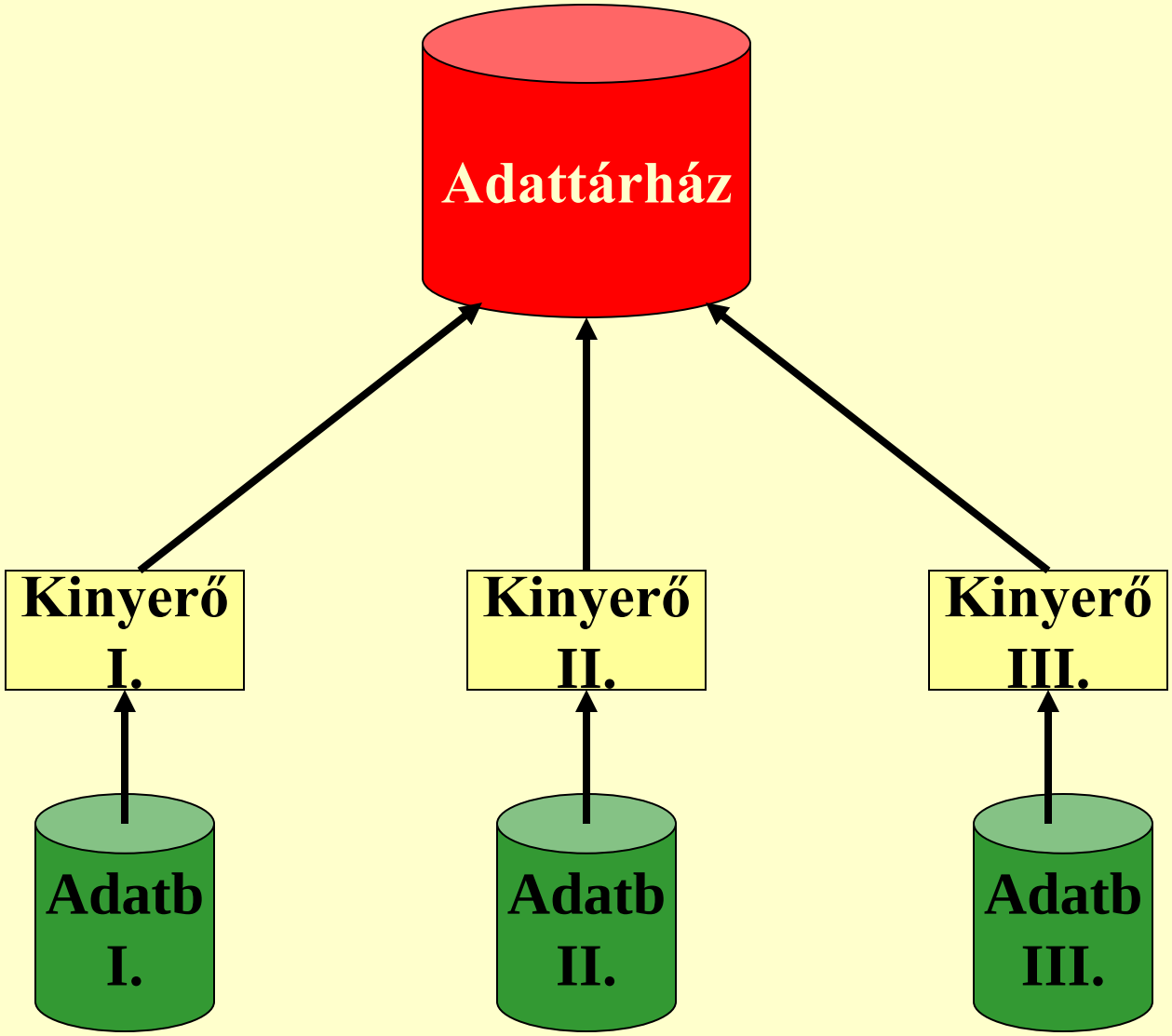


Data Warehouse: Adattárház

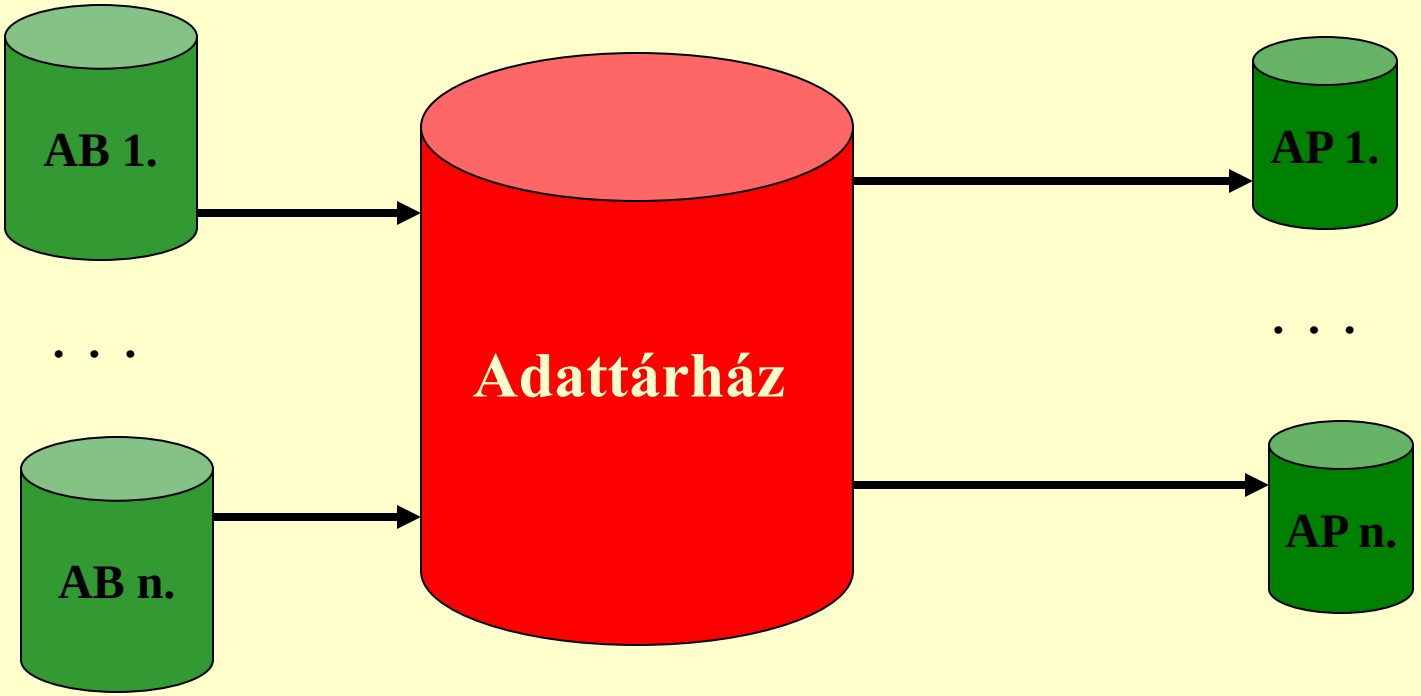
Bill Inmon:

„Az adattárház rendszer egy témaorientált, integrált, időben változó, nem átmeneti adat-rendszernek tekinthető, melynek elsődleges célja a stratégiai döntések támogatása.”

Adattárház


Adattárház
Kinyerő I.
Kinyerő II.
Kinyerő III.
Adatb I.
Adatb II.
Adatb III.

Data Marts: Adatpiacok



Operatív adatok

Adatpiacok

Adatpiacok: definíciók

1. Az adatpiac az adattárház egyik fontos komponense, egy, kiválasztott tárgyaknak osztályhoz kötött részhalmaza.

2. Az adatpiac egy alkalmazás-központú adattárház, a teljes adattárház egy része.

3. Az adattárház nem más, mint adatpiacok összessége.

Adatpiacok: előnyök

1. Adatkezelés: minden osztály maga állapíthatja meg az általa használt adatok struktúráját.
2. Relevancia: egy-egy osztály eldöntheti, hogy a historikus adatokból menynyire van szükség.
3.Önálló felhasználás: minden osztály maga döntheti el, mikor , milyen folyamatot futtat.
4. Hatékonyság: a kisebb egységek kezelése olcsóbb.

Adatpiacok: egyéb tulajdonságok

Különféle célokra sok adatpiac építhető ki egy adattárház ból, ezek akár átfedőek is lehetnek.

Egymástól függetlenül kiépült adatpiacok egy hálózatban egy virtuális adattárházat alkothatnak.

Tudásfeltárás

Definíció:
rejtett, ismeretlen, potenciálisan hasznos tudás kinyerése az adatokból nem triviális módon.

Több tudományágat átfogó kutatási terület.

Adatbányászat: az adatok összefüggéseinek feltárása.

Lépések

- **adatkiválasztás**
- **adattisztítás**
- **bővítés**
- **szűkítés**
- **kódolás**
- **adatbányászat**
- **jelentéskészítés**

Minta feladat

Egy kiadó magazinokat árul.
Kapcsolatokat keresünk az előfizetők tulajdonságai és előfizetési szokásai között.

Adatkiválasztás

Kiválasztjuk a szükséges adatokat az operatív adatbázisokból.

Pl.
ügyfélszám
név
cím
előfizetési dátum
magazin neve

Tisztítás

- **Véletlen kettőződések**
- **Név elírások (Kotsis, Kocsis, Kotsits)**
- **Kitöltés hiánya (alapértelmezés)**

Bővítés

Hozzáveszünk újabb adatokat:

- **születési idő**
- **jövedelem**
- **van-e autója**
- **van-e háza**
- **stb.**

Szűkítés

Kihagyhatunk sorokat:
pl. nincs kitöltve (?)

Kihagyhatunk oszlopokat:
pl. a név nem kell már (a tisztításhoz kellett!)

Jól meggondolni, mert elveszíthetünk fontos információkat!

Kódolás

Főként, ha az adat „túl részletes.

Pl.
Születési dátum helyett: korkategória
Cím helyett: régió kód
Előfizetés kezdete helyett: időtartam hónapokban
Jövedelem helyett: ezerrel(?) osztva
stb.

Alapvetően befolyásolhatja az eredményt!

Adatbányászat

Sokféle technika lehet, néhány ezek közül:

hagyományos lekérdező eszközök
statisztikai technikák
vizuális technikák
hasonlóság, távolság, szomszédság
döntési fák
társító szabályok
stb.

Hagyományos lekérdező eszközök

Egyszerű lekérdezések: pl. átlagszámítások.

Ezek szabhatják meg a további lépéseket!

(Pl. ha egy magazint 10 % vásárol, egy „90%-osan jó” módszer azt mondhatja, hogy ez nem kell senkinek!)

Statisztikai technikák

Összefüggéseket keresünk:
kor és vásárolt magazin
két magazint vásárlók tulajdonságai
stb.

Vigyázat! I.

Csak az olyan összefüggés nyereség, ami nem triviális:
Pl. nem meglepő, ha egy konkrét újságnál is ugyanazt az eloszlást tapasztaljuk a vásárlók életkora szerint, ami általában igaz.

Vigyázat! II.

Estleg a statisztikában mutatkozik olyan halmaz, aki semmilyen magazinra sem fizet elő. Ez adatszennyeződés, érdemes megvizsgálni, mi lehet az oka.

Vizuális technikák

Mivel nem tudjuk mit keresünk, segíthet, ha a számítógép ábrázoljuk a különféle eloszlásokat, esetleg időben változva. Érdekes összefüggéseket vehetünk észre.

Hasonlóság és távolság

A rekordokat tekinthetjük egy n dimenziós tér pontjainak. Közöttük lehet távolságot definiálni. (Legegyszerűbb az Euklidesi távolság:

$$\sqrt{(x_1-y_1)^2+\ldots(x_n-y_n)^2}$$

Érdekes csoportokat találhatunk. Kérdés: hány dimenziót vizsgáljunk? (És melyek legyenek azok?) Projekció!

Megjegyzés

Az OLAP eszközök is ilyenek, használhatóak is az adatbányászatban, de azok rögzített kérdésekre adnak választ.

Vigyázat!

A távolság numerikus értékét meghatározza a kódolás!
Pl.: a jövedelem nagyságrendje más, mint a koré: a fontossága is más lesz!

Szomszédság

A k db. legközelebbi szomszéd viselkedése adhat előrejelzést a vizsgált objektum viselkedésére.

Vigyázat!

A túl egyenletes eloszlásnál nem mond semmit, nincsenek jellegzetes osztályok.

Túl sok dimenzió esetén nem lesznek jellegzetes osztályok.

Új problémák

Nagyon sok a rendelkezésre álló adat:
a hálózat maga egy adatbázis.
Válogatás kell (felesleges másolat, elektronikus szemét).
Keresés (általában túl primitív, sok találat).
Ügynök (agent) programok (barátságos, barátságtalan).
Kiszolgáltatottság!

Vége