1. dia

Dr. Kotsis Domokos

Adatbázisok

Segédanyag az előadáshoz

1

2. dia

Cél

Cél: információ tárolás, feldolgozás, továbbítás.

2

3. dia

Az információ feldolgozás alapvető módszerei

- Szóbeli
- Írásbeli
  Nyomtatott
- Elektronikus

3

4. dia

A szóbeli információ feldolgozás

Az információ tárgya akkor, ott nem kell, hogy jelen legyen.

Szóbeli leírás: magas absztrakciós szint.
Interaktív.

4

5. dia

Mit kell tudni?

Forrás: **értelmes beszéd.**
Nyelő: **a beszéd értése.**

Kicsi különbség.

5

6. dia

Az írásbeli információ feldolgozás

Az információ szolgáltatás tárgyán kívül annak forrása (az információ „adója") sem kell, hogy akkor, ott jelen legyen.

Nincs metakommunkáció, nincs interakció : még magasabb absztrakciós szint.
Hang nincs, de képek, utalások lehetnek.
Nyomtatás: sokakhoz eljut.

6

7. dia

Mit kell tudni?

Forrás: írás tudás, eszközök használata
(stilus, lúdtoll, töltőtoll, írógép).
Nyelő: olvasás tudás.

Nagyobb különbség.

7

8. dia

Az elektronikus információ feldolgozás

Az eddigieken kívül:
A korábbi módszerek határai kitágulnak (telefon, Email).
Az információ szolgáltatásnak nem egy forrása van (rádió televízió, internet).

Változatos hangok, képek (mozgók is), interaktív.
Alacsonyabb absztrakciós szint.
Mindenkihez eljuthat.

8

9. dia

Mit kell tudni?

Forrás: tv, rádió műsorok készítése;
távközlési hálózat tervezése, üzemeltetése;
számítógépek hálózatok programozása.

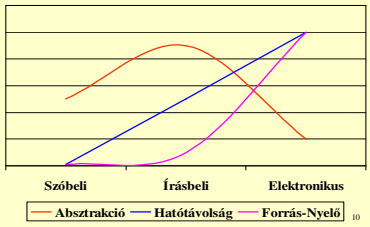
Nyelő: tv, rádió, telefon kezelése; számítógépek,
programok használata (ECDL).

Óriási különbség

9

10. dia

Összefoglalás



10

11. dia

Szakmai bevezetés

Az információfeldolgozás kezdetei: a Hollerith gépek
A számítógépek kezdeti felhasználási területei. Számítástechnika-Informatika.
A mai helyzet: információ = energia?

11

12. dia

Az információ feldolgozás alapvető módszerei

- **Folyamat szemléletű információ feld.**
- **Adatbázis szemléletű információ feld.**
- **Döntés-támogató rendszerek.**
- **Tudásbázisok (szakértői rendszerek).**

12

13. dia

A folyamatszemléletű információ feldolgozás

cél → folyamat → állományok

Előnyök:
- csak a szükséges adatokkal dolgozik
- optimális adatstruktúra használható

Hátrányok:
- többszörös tárolás (inkonzisztencia)
- csak egy cél

13

14. dia

Optimális struktúra

Opt. szemp.
- tárolóhely
- feldolgozási idő
  - létrehozás
  - keresés
  - változtatás
- felhasználás egyszerűsége

14

15. dia

Keresés 1.

Egy N elemű file-ban egy rekord keresésének lépései:

$$S_N \quad (A+B+C)$$

A keresési idő:

$$T_N = \sum_{i=1}^{N} p_i \sum_{j=1}^{S_i} (A_{i,j}+B_{i,j}+C_{i,j})$$

ahol p_i az i-edik elem keresésének valószínűsége ($\sum_{i=1}^{N} p_i = 1$),

és S_i az i-edik elem megtalálásához szükséges lépésszám.

15

16. dia

Keresés 2.

Fontos a hardware, a tároló szerkezete!

RAM: a címek sorrendje közömbös.

Mozgófejes lemez: pozícionálás, forgás, beolvasás.

16

17. dia

Keresés 3.

átlagos t időt véve:

$$T_N = \sum_{i=1}^{N} p_i \, t$$

egyenlő gyakorisággal számolva:

$$T_N = S_N t$$

17

18. dia

A keresést meghatározó tényezők

Struktúra

Algoritmus

Pl. rendezetlen esetben lineáris keresés:

$S_N \approx N/2 \quad S'_N \approx N$;

ugyanez rendezett esetben:

$S_N \approx N/2 \quad S'_N \approx N/2$;

de itt nyilván van jobb módszer is.

18

19. dia

A legfontosabb állomány struktúrák

- Szekvenciális állomány struktúrák
- Hierarchikus állomány struktúrák;
- Hálós állomány struktúrák;
- Nem konzekutív állomány struktúrák

19

20. dia

Fizikai szekvenciális állomány struktúrák

A rekordok logikai és fizikai sorrendje megegyezik.

Keresési módok:

Lineáris keresés

Bináris, logaritmikus keresés

Peterson-féle keresés

20

21. dia

Bináris, logaritmikus keresés

Minden lépésben hosszra nézve felezzük a vizsgálandó állományt.

$$S_N \approx S'_N \approx \log_2(N)$$

Heurisztikus bizonyítás:

legyen K olyan, hogy $N=2^K$.

Minden felezésnél $K_{új} = K_{régi} - 1$.

$K_{új}=0$ éppen innen K lépés után lesz, és

$$K=\log_2(N)$$

21

22. dia

Peterson-féle keresés

Minden lépésben az előfordulás valószínűségére nézve felezzük a vizsgálandó állományt.

Egyenletes eloszlás mellett:

$$A = A_E + \frac{A_U - A_E}{K_U - K_E} * K$$

ekkor:

$$S_N \approx S'_N \approx \tfrac{1}{2} \log_2(N)$$

22

23. dia

Bináris vs. Peterson keresés

S_N a Petersonnál kisebb, de

1. az eloszlás nem feltétlenül egyenletes,
2. Petersonnál valódi osztás kell, a binárisnál a léptetés alkalmazható.

A leggyorsabb, ha $N = 2^k - 1$ alakú.

Ha nem ilyen:

1. két részre osztás
2. kiegészítés álrekordokkal

23

24. dia

A fizikai szekvencialitás előnye, hátránya

A keresés lépésszáma kicsi, de a rendezettség biztosítása is időigényes!

1. bonyolult beszúrási algoritmus
2. rendszeres fizikai rendezés

24

25. dia

Logikai szekvenciális állomány struktúrák

A rekordok logikai és fizikai sorrendje nem egyezik meg.

Mutatórendszerek (egy- kétirányú, gyűrűs).

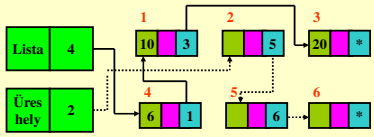
Indító tábla.

Dinamikus file-oknál jó: csak mutatókkal kell dolgozni.

25

26. dia

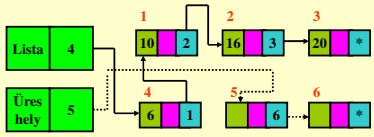
Példa



26

27. dia

Beszúrás

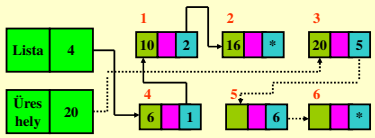


Új elem: 16

27

28. dia

Törlés



Törlendő: 20

28

29. dia

Hátrányok

A fizikainál ismert keresési algoritmusok nem használhatók.

Mozgófejes lemeznél:

Pozícionálás, forgási idő kell az eléréshez.

Véletlenszerű elhelyezkedésnél C cilindernyi anyagnál átlagosan C/3 cilindernyi pozícionálás kell.

29

30. dia

Csaknem fizikai szekvenciális file

Létrehozáskor fizikai szekvenciálisan elhelyezett logikai szekvenciális file.

Feldolgozáskor logikai szekvenciális.

Túlcsordulási terület a cilinderen.

30

31. dia

Kupacos keresés 1.

N rekord, G rekord/ csoport, N/G csoport.

$$\sum_{k=1}^{\frac{N}{G}} k\left(\frac{1}{\frac{N}{G}}\right) = \frac{\frac{N}{G}+1}{2}$$

31

32. dia

Kupacos keresés 2.

$$\overline{S}_N \approx \frac{\frac{N}{G}+1}{2} + \frac{G-1}{2} = \frac{1}{2}\frac{N}{G} + \frac{G}{2}$$

32

33. dia

Kupacos keresés 3.

A minimum kereséshez G szerint deriválva:

$$\frac{N}{G^2} = \frac{1}{2}$$

Ennek minimum helye: $G = \sqrt{N}$

Ebből következően: $\overline{S}_N \approx \sqrt{N}$

33

34. dia

Gyakoriság szerint rendezett file

Statikus, dinamikus megoldás.

Önrendező algoritmusok.

34

35. dia

Hierarchikus struktúrák 1.

Egy megelőző, több rákövetkező (fa).

Nyelvi leírás: COBOL, LISP

35

36. dia

Hierarchikus struktúrák 2.

Ábrázolás:

- **A.) Belső mutatós módszerek**
  1. **Left-list**
  2. **Több pointer**
  3. **Segédrekordok**
  4. **Gyűrűk**

36

37. dia

Hierarchikus struktúrák 3.

B.) Külső mutatós módszerek
- **1. Táblázatok**
- **2. Bináris mátrixok**

37

38. dia

Hálós struktúrák 1.

1. Visszavezetés hierarchikusra.
2. Az ott leírt módszerek?
A.) Belső mutatós módszerek
- **1. Left-list nem**
- **2. Több pointer igen**
- **3. Segédrekordok nem**
- **4. Gyűrűk nem**

38

39. dia

Hálós struktúrák 2.

B.) Külső mutatós módszerek
- **1. Táblázatok igen**
- **2. Bináris mátrixok igen**

39

40. dia

Nem konzekutív struktúrák

Indexelt szervezés

Direkt szervezés

40

41. dia

Indexelt szervezés

1. Sűrű indexelés

2. Ritka indexelés

41

42. dia

Sűrű indexelés

Minden rekordhoz tartozik index bejegyzés

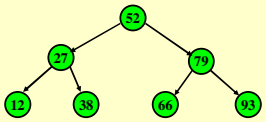
Előnyök: több index szervezhető
Hátrányok: nagyok az index file-ok

Mikor lehet gyors a keresés?

42

43. dia

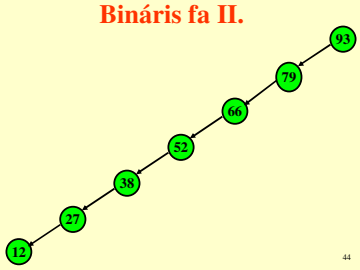
Bináris fa I.



43

44. dia

Bináris fa II.



44

45. dia

B fák

R. Bayer 1970

n a B fa rendje

1. **Minden lap legfeljebb 2n tételt tartalmaz.**
2. **Minden lapon – a gyökér kivételével - legalább n tétel van .**
3. **Ha egy lapon m tétel van, vagy levéllap, vagy m+1 utódja van.**
4. **Minden levéllap ugyanazon a szinten van.**

45

46. dia

B fák elérése

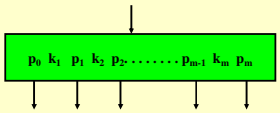
A kereséshez szükséges átlagos lépésszám a laphivatkozásoktól függ, ezek száma N elemnél n rendű fában:

$$\log_n N$$

46

47. dia

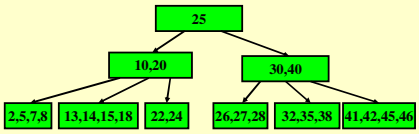
B fa egy lapja



47

48. dia

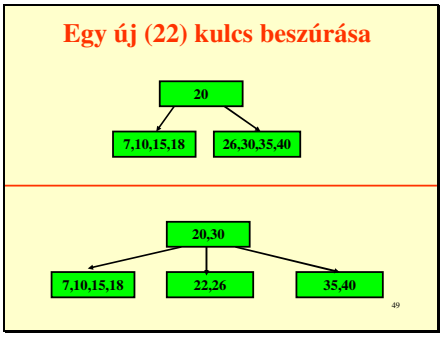
Másodrendű B fa



48

49. dia

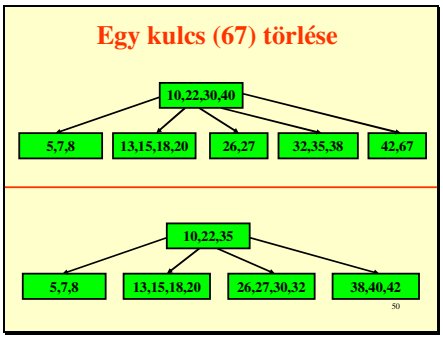
Egy új (22) kulcs beszúrása



49

50. dia

Egy kulcs (67) törlése



50

51. dia

B+ fák

A gyakorlatban index állományok készítésére az u.n. B+ fákat használják. (Ilyen struktúrát alkalmaz – sok más rendszerrel együtt – az Oracle is.)

A B+ fa egy olyan B fa, melyben azok a pointerek, melyek adatokra (adatok címére) mutatnak, s amelyeket keresünk, valamennyien a levéllapokon helyezkednek el, így a levéllapok és a többi lap struktúrája különbözik.

51

52. dia

Ritka indexelés

„Jelző cölöpök” mutatják a rekord helyét.
Rendezettség kell!
Több szintű index rendszer is készíthető.
Gyors elérés, de csak egy szempont szerint.

52

53. dia

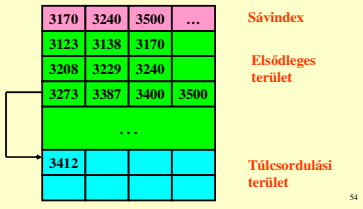
Index-szekvenciális szervezés 1.

A ritka indexelés egy megvalósítása.
A fizikai architektúrának megfelelően:
Sáv; cilinder; fő index.
Index; elsődleges; túlcsordulási terület.
Szervezés: az elsődleges területen fizikai, a túlcsordulási területen logikai szekvenciális szervezés.

53

54. dia

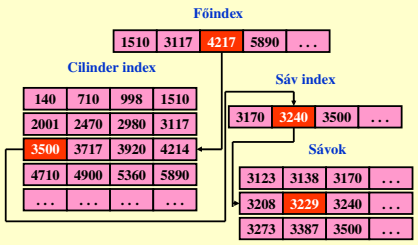
A tároló felosztása



54

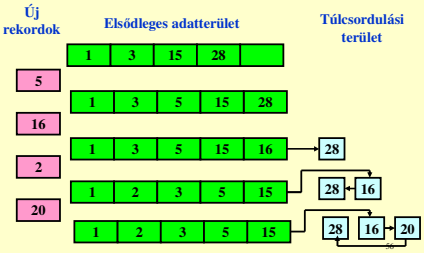
55. dia

Egy rekord (3229) megkeresése



56. dia

Rekordok beszúrása



57. dia

Direkt szervezés 1.

Egy leképezést adunk meg a rekord kulcsa (K) és a címe (A) között.

$$A = f(K)$$

Kívánatos lenne az egy-egy értelműség:

$$K_1 \neq K_2 \Rightarrow f(K_1) \neq f(K_2)$$

57

58. dia

Direkt szervezés 2.

Közvetlen leképezés: igen rossz a tárkihasználás.
Hashing algoritmusok: megengedjük, hogy

$$K_1 \neq K_2$$

esetén is – lehetőleg ritkán - előfordulhasson

$$f(K_1) = f(K_2)$$

58

59. dia

Két hashing algoritmus

Maradék módszer: M modulus mellett

$$A = R\left(\frac{K}{M}\right)$$

Csonkítás: a kevés információt hordozó jegyek elhagyása.

59

60. dia

Szinonímok

A hashing algoritmusok nem egy-egy értelműek, így szinonimok keletkezhetnek.
Szinonim kezelő módszerek:
Külső láncolás
Belső láncolás
Nyílt (Peterson) módszer
Többszörös hashing
Bucket-ek használata

60

61. dia

A külső láncolás

A szinonimokat mutatók kötik össze.

Külön túlcsordulási terület.

Problémák:

- **Tárkihasználás**
- **Törlés: az elsőt nem lehet törölni.**

61

62. dia

A belső láncolás

A szinonimokat itt is mutatók kötik össze, de azok az (egyetlen) elsődleges terület még nem használt helyeire kerülnek, az eredeti címhez a lehető legközelebb.

A tárkihasználás jó, de másodlagos szinonimok keletkezhetnek.

62

63. dia

A nyílt módszer

A rekordok ugyanúgy helyezkednek el, mint a belső láncolásnál, de nincsenek mutatók.

A keresésnél szükséges lépésszám nagyobb, mint a belső láncolásnál.

63

64. dia

A többszörös hashing

Ha a leképezés szinonimra vezet, egy másikkal próbálkozunk.

(A gyakorlatban két hashing algoritmust, majd a korábban ismertetett módszerek valamelyikét használják.)

64

65. dia

Bucket-ek használata

E módszernél nem egy-egy rekordnak, hanem egy-egy rekord csoportnak van külön címe.

Kevesebb szinonimkezelésre van szükség, de a csoportokon belül is kell keresnünk.

Hasznos akkor lehet, ha hardware szempontok (szektor) indokolják.

65

66. dia

Példa szinonímkezelésre

35, 46, 18, 14, 63, 06,75, 85,91, 52.

Leképezés: a 11-el osztás maradéka.

Második: a 7-el osztás maradéka.

66

67. dia

Példa bucket használatra

21, 32, 71, 14, 83, 72, 43, 51, 02, 11, 93, 44.

Leképezés: 4 bucket-be a második jegy szerint.

67

68. dia

Összehasonlítás 1.

Az egyes módszerekhez átlagosan szükséges lépésszámot a tényleges (N), és az egyáltalán elhelyezhető (M) rekordszámmal meghatározott A=N/M u.n. kitöltési tényező függvényében vizsgáljuk.

68

69. dia

Összehasonlítás 2.

Belső láncolásnál sikeres esetben:

$$S_N = 1 + \frac{1}{8A}\left(e^{2A} - 1 - 2A\right) + \frac{1}{4}A$$

Sikertelen esetben:

$$S_N^{'} = 1 + \frac{1}{4}\left(e^{2A} - 1 - 2A\right)$$

69

70. dia

Összehasonlítás 3.

Nyílt módszernél sikeres esetben:

$$S_N = \frac{1}{2}\left(1 + \frac{1}{1-A}\right)$$

Sikertelen esetben:

$$S_N' = \frac{1}{2}\left[1 + \left(\frac{1}{1-A}\right)^2\right]$$

70

71. dia

Összehasonlítás 4.

Többszörös hashing-nél sikeres esetben:

$$S_N = -A^{-1}\ln(1-A)$$

Sikertelen esetben:

$$S_N' = (1-A)^{-1}$$

71

72. dia

Összehasonlítás 5.

Példaképpen sikeres esetben:

$$1 + \frac{1}{8A}\left(e^{2A} - 1 - 2A\right) + \frac{1}{4}A = 1 + \frac{1}{2}A + \sum_{i=2}^{\infty}\frac{2^{i-2}A^i}{(i+1)!}$$

$$\frac{1}{2}\left(1 + \frac{1}{1-A}\right) = 1 + \frac{1}{2}A + \frac{1}{2}A^2 + \ldots = 1 + \frac{1}{2}\sum_{i=1}^{\infty}A^i$$

$$-A^{-1}\ln(1-A) = 1 + \frac{A}{2} + \frac{A^2}{3} + \ldots = \sum_{i=0}^{\infty}\frac{A^i}{i+1}$$

72

73. dia

Összehasonlítás 6.

Sikertelen esetben:

$$1+\frac{1}{4}\left(e^{2A}-1-2A\right)=1+\frac{1}{4}\sum_{i=2}^{\infty}\frac{(2A)^i}{i!}$$

$$\frac{1}{2}\left[1+\left(\frac{1}{1-A}\right)^2\right]=1+A+\frac{3}{2}A^2+\ldots=1+\sum_{i=1}^{\infty}\frac{i+1}{2}A^i$$

$$(1-A)^{-1}=1+A+A^2+\ldots=\sum_{i=0}^{\infty}A^i$$

73

74. dia

Összehasonlítás 7.

Nem elég a lépésszámot vizsgálni, hisz az egyes lépések időigénye módszerenként más.

Figyelembe kell venni a hardware adottságokat is.

74

75. dia

Összehasonlítás 8.

Gyors véletlen elérésű (pl. RAM) tárolónál:

TH: nagy algoritmusidő.

NyM: több lépés mint BL-nél.

BL: másodlagos szinonimok miatt több lépés, mint KL-nél.

KL: leggyorsabb, de rossz tárkihasználás.

75

76. dia

Összehasonlítás 9.

Mozgófejes tárolónál:

TH, KL: sok fejmozgás.
NyM: több lépés mint BL-nél.

De NyM-hez nem kell CPU, így a leggyorsabb lehet.

76

77. dia

Alapelv

A rendelkezésre álló adatokból indulunk ki, az „összes" adatot (entity) és a közöttük fennálló „összes" kapcsolatot (relationship) egy integrált adatbázisba gyűjtjük, és valamennyi felhasználó valamennyi kérdéséhez ezt az adatbázist (vagy ennek egy részét) használhatja.

77

78. dia

Adatreprezentáció

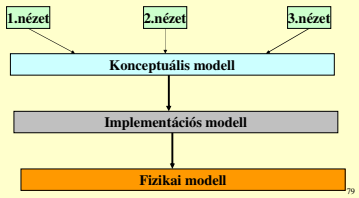
Az „összes adat" egy „mini világra" vonatkozik.
Ennek leírása több szintű:
- ***konceptuális*** **(magas szintű),**
- ***implementációs*** **(reprezentációs),**
- ***fizikai*** **modell.**

Az egyes felhasználói *nézetek* (view-k) az adatbázisnak csak egy részét látják.

78

79. dia

A leképezések (mapping-ek)



79

80. dia

Egyed-kapcsolat leírások

Az alapvetően használt modell az *egyed-kapcsolat* (entity-relationship), azaz ER modell.
Az egyedeknek *tulajdonságaik* (attribute) vannak, ezek egyike, vagy ezekből egy készlet az egyedet egyértelműen azonosító *kulcs*.
A kapcsolatok lehetnek 1:1, 1:n, m:n jellegűek.
A kapcsolatokat sokszor egyedhalmazzá alakítjuk (kapcsoló rekord: a kapcsolat mint egyed).

80

81. dia

Adatbázis felügyelő

Több felhasználó: védeni kell tőlük a rendszert, és őket egymástól.
Ez az *adatbázis felügyelő* (data base administrator) egyik feladata.

Tervezés: *séma, alséma* készítés.
Eszköz: DDL.

81

82. dia

Adatfüggetlenség

***Logikai*: az adatok logikai struktúrájában (konceptuális, implementációs modell) történő változtatás ne érintse az ezen változtatást nem igénylő felhasználók munkáját.**

***Fizikai*: a fizikai modell változtatása ne kényszerítse ki az implementációs modell változtatását.**

A kettő együtt az *adatfüggetlenség*.

82

83. dia

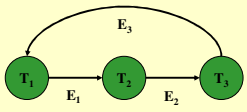
Tervezési szempontok

Kapcsolat múlttal, jövővel.
Biztonság, titkosság, pontosság.
Konkurrens folyamatok kezelése.
Válaszidő.
Kényelmes használat.
Költségek.

83

84. dia

Várakozási gráf



84

85. dia

Az ABF üzemeltetési feladatai

Mentések, karbantartás.
Kapcsolattartás a felhasználóval.

85

86. dia

A DML

A felhasználó eszköze.
Többféle felosztás:

1. **Alkalmazott nyelv szerint:**
   - **Host language**
   - **Self Contained Language**
2. **A felhasználás jellege szerint:**
   - **Procedurális**
   - **Deklaratív**

86

87. dia

Host Language (Beépített, beágyazott nyelvű) rendszerek

Egy korábban ismert magas szintű nyelvet használnak.

Megoldások:

- **Eljárás gyűjtemények könyvtárban.**
- **Eljárás gyűjtemény előfordítóval.**
- **Eljárás gyűjtemény interpreterrel.**

87

88. dia

Self Contained (Önálló) nyelvű rendszerek

Lehet külön programozható nyelv.
Lehet, csak paraméterezendő rendszer.
Lehet 4GL fejlesztő rendszer.

88

89. dia

Procedurális rendszerek

Egy lépésben egy rekordot dolgoznak fel (record-in-time feldolgozás).

89

90. dia

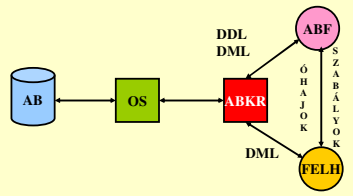
Deklaratív rendszerek

Egy lépésben egy meghatározott tulajdonságú halmazt (pl. egy táblázatot) dolgoznak fel(set-in-time rendszerek: ilyen pl. az SQL).

90

91. dia

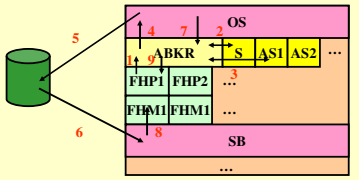
Az OS és az ABKR



91

92. dia

Egy lekérés folyamata



92

93. dia

Alapvető ABKR modellek (approach)

1. Hierarchikus
2. Hálós (CODASYL, DBTG)
3. Relációs

93

94. dia

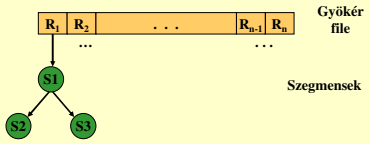
A hierarchikus modell I.

A legelső modell, ma már nem használják.
Az adatokat fákban tároljuk, ahol a fák egy-egy szögpontja a *szegmens* (segment) adatokat és a további szegmensekre utaló mutatókat tartalmaz.
A fák gyökérelemei hagyományos állományokba vannak szervezve. Az egyes nézetek a számukra *érzékeny szegmenseket* (sensitive segment) látják.
Jellegzetes képviselője az IMS (IBM).

94

95. dia

A hierarchikus modell II. (IMS)



95

96. dia

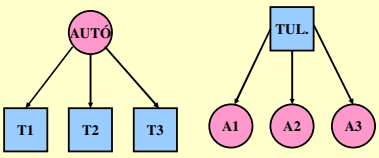
A hierarchikus modell tulajdonságai

Igazi ABKR, hisz többfelhasználós, az egyes felhasználók nem ugyanazt látják.
Nem igazi ABKR, mert a lekérdezés hatékonysága erősen függ az adatstruktúrától.

96

97. dia

Lehetséges struktúrák egy hierarchikus modellben



97

98. dia

A hálós modell 1.

A CODASYL bizottság által létrehozott DBTG (Data Base Task Group) jelentései (1969-1971) hozták létre.
Két évtizedig szinte kizárólag ezt használták.
Terminológiai és metodológiai javaslatokat tartalmaz.

98

99. dia

A hálós modell 2.

Már bevezetett fogalmak:

- **DDL**
- **DML**
- **séma**
- **alséma.**

99

100. dia

A hálós modell 3.

Új fogalmak:

Area: valamilyen szempontból egységesen kezelendő adathalmaz.

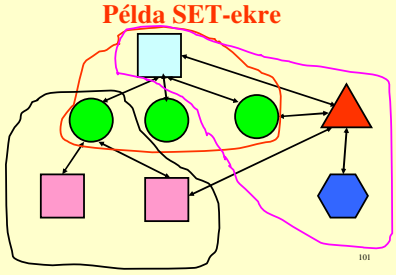
Set: kétszintű fa, melynek gyökéreleme a *tulajdonos* (owner), levelei a *tagok* (members).

A set-ek segítségével a legbonyolultabb hálós kapcsolatok is leírhatók.

100

101. dia

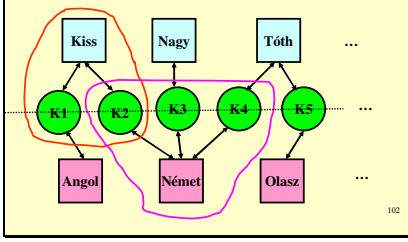
Példa SET-ekre



101

102. dia

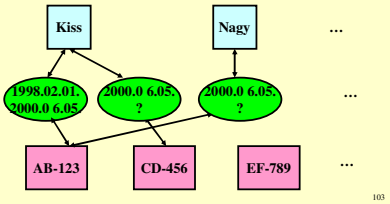
Kapcsolatok hálós modellben 1.



102

103. dia

Kapcsolatok hálós modellben 2.



103

104. dia

Séma hálós modellben

Séma név.
Zárak, kulcsok.
Rekord leírások (kalkulált mezők, kódolt mezők).
Kapcsolat leírások: SET-ek.
AREA leírások.

104

105. dia

Alséma hálós modellben

A séma leírás valódi része.
Újdonság nem szerepelhet, de mindent át lehet nevezni.

105

106. dia

Lekérdezés hálós modellben

Az eredeti javaslat a COBOL nyelvet javasolta (BATCH feldolgozás!), ezt váltotta fel a PL1, majd az interaktív felhasználás.

Fontos reprezentáns: IDMS (IBM 360 és 370: ESzR 1. és 2.).

106

107. dia

Az IDMS specialitásai

Member-ek rendezése.
Indexek készítése.
Hashing algoritmus haználata.

107

108. dia

A relációs modell 1.

A relációs modell ötlete Codd 1970-es cikkéből származik, így lényegében egyidős a DBTG jelentéssel.
Két évtized után átvette a vezető szerepet, ma szint kizárólag ezt használják, ezért ezt vizsgáljuk részletesen.

108

109. dia

A relációs modell 2.

A *reláció* ebben az értelemben tulajdonképpen egy *táblázat* (table).
A táblázat oszlopai a *tulajdonság*ok (domains), a sorai az *n-es*ek (tuples).
Gyakran nevezik azonban a sorokat rekordnak, az egyes oszlopokhoz tartozó értékeket mezőnek (field).
Az R reláció sorainak hosszát r-el jelöljük.

110. dia

Alapfeltételek relációkkal kapcsolatban

1. **Ne legyenek teljesen megegyező tartalmú sorok vagy oszlopok,**
2. **a sorok és az oszlopok sorrendje ne hordozzon információt.**

Azt a mezőt, vagy mezőkészletet, mely a sort (a sor többi elemét) egyértelműen meghatározza, *szuperkulcs*nak nevezzük. A nem szűkíthető szuperkulcs neve *kulcs*. (1. miatt van ilyen.)

111. dia

Anomáliák

1. **Módosítási anomália**
   egy attribútum értéke több helyen szerepel, így több helyen kell módosítani (inkonzisztencia).
2. **Beírási anomália**
   hiányos adatok miatt valamit nem lehet bevinni (információvesztés).
3. **Törlési anomália**
   egy sor törlésével olyan információ is elvész, amire még szükség lenne (információvesztés).

112. dia

Példa anomáliákra

Egy reláció oszlopai:
Tantárgyszám, tantárgynév, követelmény, oktatónév, oktatócím

- **Változtassuk meg az oktató címét.**
- **Mi történik, ha egy új tárgyhoz nincs oktató kijelölve?**
- **Mi történik, ha egy oktató kilép?**

112

113. dia

Az anomáliák kiküszöbölése

A felsorolt anomáliák kiküszöbölhetők, a relációk olyan felbontásával (dekompozíció), melyek eredményei normalizált relációk.

113

114. dia

1. Normálforma

A reláció 1NF értelemben normalizált, ha minden sorának minden mező értéke egy elemi érték.

(Újabb rendszerek lehetővé teszik a figyelmen kívül hagyását.)

114

115. dia

2. Normálforma

A reláció 2NF értelemben normalizált, ha 1NF értelemben normalizált, és ha egy mezőérték meghatározásához összetett kulcs szükséges, egy másik mezőérték meghatározásához nem elegendő ennek egy része.

115

116. dia

Példa 2. normálformára

cikkszám, ár, szállítási sorszám, cikkleírás

Anomália: ha az utolsó sort töröljük, elvész a cikkszám, cikkleírás közötti összefüggés.
Ok: az ár azonosításához szükséges a cikkszám, szállítási sorszám összetett kulcs, a cikkle1rás meghatározásához elég a cikkszám.
Megoldás: dekompozíció.
cikkszám, cikkleírás
cikkszám, ár, szállítási sorszám

116

117. dia

3. Normálforma

A reláció 3NF értelemben normalizált, ha 2NF értelemben normalizált, és ha egy mezőérték sem függ olyan attribútum értéktől, vagy ezek olyan halmazától, mely nem kulcs.

117

118. dia

Példa 3. normálformára

rendszám, gyártmány, típus, gyártási idő, szín

Ha egy időben csak egy színt használnak, elvész a gyártási idő és a szín közötti összefüggés.

118

119. dia

Boyce-Codd (BCNF) normálforma

A 3NF tovább gondolása.
A reláció BCNF értelemben normalizált, ha 3NF értelemben normalizált, és ha egy összetett kulcs egy részét sem határozza meg más attribútum érték (nincs kulcstörés).
Ha egy reláció BNCF, akkor 3NF is.

119

120. dia

Példa BCNF normálformára 1.

tantárgy, tanár, diák

Tegyük fel, hogy egy tanár csak egy tárgyat oktat.
Mi legyen a kulcs?
- **Egyszerű kulcs nem egyértelmű.**
- **tantárgy, tanár nem határozza meg a diákot**
- **tanár, diák nem 2NF (a tanár meghatározza a tantárgyat).**
- **tantárgy, diák nem BCNF (a tanár meghatározza a tantárgyat).**

120

121. dia

Példa BCNF normálformára 2.

Anomália: törléskor eltűnik, hogy egy tanár milyen tárgyat tanít.

Megoldás: dekompozíció

tantárgy, tanár
tanár, diák

122. dia

Többértékű függőség

Az $A_1A_2...A_n \rightarrow\rightarrow B_1B_2B_m$ többértékű függőség fennáll, ha
az R reláció soraiból tekintve azokat, melyek minden A_i értéken megegyeznek, ezek Bj-ken felvett értékei függetlenek az A-któl és B-ktől különböző attributumok értékeitől.

A többértékű függőség nem triviális, ha
1. **egyik B sincs az A-k között,**
2. **az A-k és a B-k együttesen sem tartalmazzák R összes atribútumát.**

123. dia

4. Normálforma

Egy reláció 4NF-ben van, ha valahányszor az $A_1A_2...A_n \rightarrow\rightarrow B_1B_2B_m$ nem triviális többértékű függőség fennáll, akkor $\{A_1A_2...A_n\}$ szuperkulcs.

Ha a reláció 4NF, akkor BNCF is.

124. dia

Példa többértékű függőségre I.

Pl. legyenek az R reláció oszlopai:
Tanár Tantárgy Kar

Egy tanár több tárgyat taníthat, egy tárgyat többen taníthatnak, egy tárgyat több karon tanítanak, egy karon több tárgyat tanítanak.
Tanár→→Tantárgy és Tantárgy→→Kar fennáll, és nem triviális.

124

125. dia

Példa többértékű függőségre II.

Tanár	Tantárgy	Kar
...		
Nagy	AB	KGK
Nagy	AB	NIK
Kiss	PPT	NIK
...		

BCF, de nem redundancia mentes:
pl. egy Tanár → Tantárgy bejegyzés többször is előfordulhat.
Tanár Tantárgy nem szuperkulcs, azaz nem 4F.

125

126. dia

Példa többértékű függőségre III.

Megoldás itt is a dekompozíció:
Tanár Tanfolyam
Tanár Kar
Tárgy Kar

Vigyázat:
Tanár Tárgy
Tanár Kar
nem jó, mert elvész, hogy melyik tanár melyik karon mit tanít.

126

127. dia

További normálformák

Létezik 5NF, de kisebb jelentősége miatt nem foglalkozunk ezzel.

128. dia

A lekérdezés elve relációs rendszerekben

1. relációs algebra
2. relációs kalkulus

129. dia

A relációs algebra 1.

Alapműveletek:
1. Unió: $R \cup S$
2. Különbség: $R-S$
3. Direkt szorzat: $R \otimes S$
4. Projekció: $\Pi_{i1,i2...ik}(R)$
5. Szelekció: $\sigma_F(R)$
ahol: Az F feltétel az [i] Θ [j], és az [i] Θ c elemi kifejezések logikai műveletekkel ($\wedge$, $\vee$, $\neg$) való összekötéséből állhat. Θ tetszőleges összehasonlító operátort ($=, \neq, <, >, \leq, \geq$) jelenthet.

130. dia

A relációs algebra 2.

Következmény műveletek:

6. Metszet: R∩S

R ∩ S = R - (R - S)

7. Hányados: R ÷ S

Feltesszük, hogy r > s, és s ≠ 0.)

Legyen

T = Π $_{1,2,\ldots r-s}$ (R), továbbá

V = Π $_{1,2,\ldots r-s}$ ((T ⊗ S)-R).

Ekkor belátható, hogy

R ÷ S = T - V.

130

131. dia

A relációs algebra 3.

8. Belső kapcsolat (inner join)

8a. Feltételes kapcsolat (Θ join):

$$R * S = \sigma_{[i]\,\Theta\,[r+j]}(R \otimes S)$$
$$[i]\,\Theta\,[j]$$

8b. Természetes kapcsolat (natural join): R * S

Hasonló a feltételes kapcsolathoz de itt a szelekció feltétele az, hogy az azonos nevű oszlopokban azonos érték szerepeljen.

131

132. dia

A relációs algebra 4.

9. Külső kapcsolat (Left, vagy Right Outer join): Tképpen egy Θ join, de a szelekció valamelyik relációból azokat a sorokat is meghagyja, melyekhez a másikban nem tartozik érték (itt a NULL érték fog szerepelni).

9a. Left Outer Join

A baloldali reláció minden sora legalább egyszer szerepel.

132

133. dia

A relációs algebra 5.

9b. Right Outer Join
A jobboldali reláció minden sora legalább egyszer szerepel.

9c. Full Outer Join
Egy Left- és egy Right Outer Join uniója.

133

134. dia

Példa relációs algebrai műveletekre 1.

Számoljuk ki az R ÷ S művelet eredményét!

R =

a	b	c	d
a	b	e	f
b	c	e	f
e	d	c	d
e	d	e	f
a	b	d	e

S =

c	d
e	f

134

135. dia

Példa relációs algebrai műveletekre 2.

Számoljuk ki az $R *_{B<D} S$ művelet eredményét!

R =

A	B	C
1	2	3
4	5	6
7	8	9

S =

D	E
3	1
6	2

135

136. dia

Példa relációs algebrai műveletekre 3.

Számoljuk ki az R * S művelet eredményét!

R =

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

S =

B	C	D
b	c	d
b	c	e
d	d	b

136

137. dia

Példa relációs algebra alkalmazására 1.

Tekintsük az alábbi három relációt.
a.) Autók: jele A
Mezői: rendszám, gyártmány, típus, szín, ...
b.) Emberek: jele E
Mezői: szemszám, név, foglalkozás, cím, ...
c.) Kapcsolat: jele K
Mezői: forgrsz, szemszám.

Feladat: **keressük ki a sárga opelek tulajdonosainak foglalkozását.**

137

138. dia

Példa relációs algebra alkalmazására 2.

A megoldás:

R1 = $\sigma_{\text{szín =sárga} \wedge \text{típus = opel}}$ (A)
R2 = $\Pi_{\text{rendszám}}$ (R1)
R3 = R2 * K (rendszám=forgrsz)
R4 = $\Pi_{\text{szemszám}}$ (R3)
R5 = R4 * E
R6 = $\Pi_{\text{foglalkozás}}$ (R5)

138

139. dia

Példa relációs algebra alkalmazására 3.

A megoldás egy lépésben:

$$\Pi_f(\Pi_{szsz}(K^*_{fsz=rsz}(\Pi_{rsz}(\sigma_{sz=sárga \wedge t=opel}(A)))) * E)$$

(foglalkozás→f, szemszám →szsz, forgrsz →fsz, rendszám →rsz, szín →sz, típus →t helyettesítéssel).

140. dia

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.

141. dia

A relációs kalkulus 2.

Atomi formulának, vagy atomnak nevezzük, az alábbi kifejezéseket:

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

2.) $s[i] \Theta u[j]$ **(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.) $s[i] \Theta c$ **(ahol c egy konstans).**

142. dia

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.

142

143. dia

A relációs kalkulus 4.

4.) A (∀s)(ψ) alakú (minden s olyan, hogy ψ igaz) kifejezések is formulák.

5.) A kifejezések a precedencia (Θ, ∃, ∀, ¬, ∧, ∨) megváltoztatása érdekében zárójelezhetők.

6.) Más formula nincs.

143

144. dia

A relációs algebra és kalkulus összehasonlítása 1.

A relációs algebra alapműveletei kifejezhetők a relációs kalkulus eszközeivel.

Unió: {t | R(t) ∨ S(t)}

Különbség: {t | R(t) ∧ ¬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 | (R(t) ∧F }

144

145. dia

A relációs algebra és kalkulus összehasonlítása 2.

{t | ¬ 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.

145

146. dia

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.

146

147. dia

Lekérdezés relációs rendszerekben 1. (ISBL)

Korai IBM termék.
Lényegében a relációs algebra kifejezéseinek linearizálása.
Nem felhasználóbarát.

147

148. dia

Lekérdezés relációs rendszerekben 2. (QBE)

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

148

149. dia

Példa QBE-re 1.

AUTÓK

rendsz.	gym.	szín	. . .
	OPEL	SÁRGA	

149

150. dia

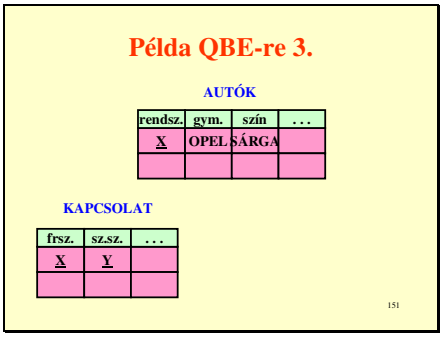
Példa QBE-re 2.

AUTÓK

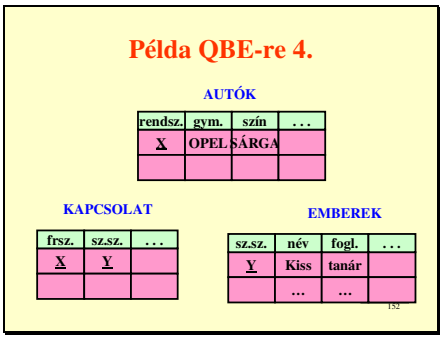
rendsz.	gym.	szín	. . .
X	OPEL	SÁRGA	

150

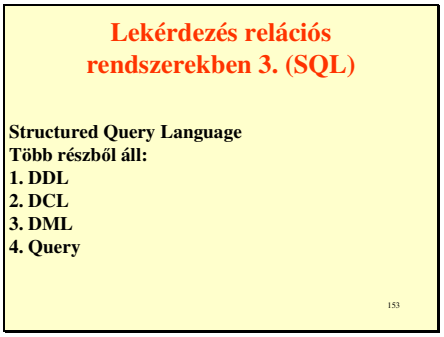
151. dia



152. dia



153. dia



154. dia

SQL verziók

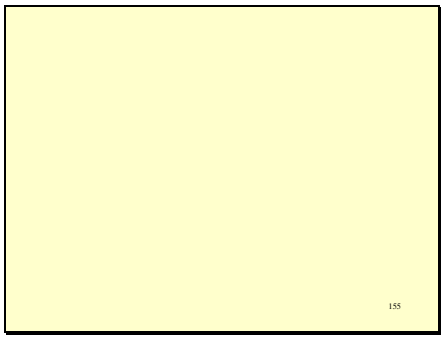
1989 SQL1

1992 SQL2
- **Hibakódok**
- **Adattípusok**
- **Adatbázis struktúrák**
- **Rendszertáblák**
- **Programozható felület**
- **Portabilitás**

Az SQL3-ról később lesz szó.

154

155. dia



155

156. dia

DDL

CREATE TABLE
ALTER TABLE
DROP TABLE

CREATE VIEW
DROP VIEW

CREATE [UNIQUE] INDEX
DROP INDEX

156

157. dia

DDL példa I.

Create Database kutya;

…

Drop Database kutya;

157

158. dia

DDL példa II.

CREATE TABLE táblanév (oszlopnév
oszlopleírás , …

esetleg index definíció
)

158

159. dia

Kulcsok legfontosabb jelzői

Táblánként max egy mező lehet

- **PRIMARY KEY**
  egyedi, nem lehet NULL értékű.
  Gyakran
  egy (előjel nélküli) egész.

- **AUTO_INCREMENT**
  automatikusan nő.

159

160. dia

Minta tábla létrehozása

```
CREATE TABLE allatok (
nev varchar(25),
fajta varchar(20),
szex char(1),
szuld date,
hald date );
```

161. dia

```
ALTER TABLE allatok
ADD COLUMN
sorszam Integer Unsigned Primary Key
   Auto_Increment FIRST
```

161

162. dia

DCL

Tranzakció kezelés:
COMMIT
ROLLBACK
LOCK
UNLOCK

Jogosítvány kezelés:
GRANT …. WITH GRANT OPTION
REVOKE

162

163. dia

DCL példa I.

GRANT jogok
ON adatbázisnév[,táblanév] TO felhasználó@ host IDENTIFIED BY ” jelszó”
WITH GRANT OPTION

164. dia

DCL példa II.

REVOKE
jogok [, GRANT OPTION] ON adatbázisnév[.táblanév] FROM felhasználó@ host

165. dia

Tranzakciók

BEGIN WORK-el kezdődik,
COMMIT-al vagy
ROLLBACK-al végződik.

166. dia

DML

INSERT
UPDATE
DELETE

166

167. dia

Insert

```
INSERT INTO allatok VALUES
('Sajti', 'eger','n','2004-02-02',NULL);
```

167

168. dia

Update

```
UPDATE allatok
SET
nev='kukac'
WHERE nev = 'giliszta'
```

168

169. dia

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*****]]**

170. dia

Select

SELECT nev, szuld FROM allatok
WHERE faj ='kutya' OR faj ='macska'
ORDER BY szuld DESC

170

171. dia

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.

171

172. dia

Beágyazott (embedded) SQL I.

1. EXEC SQL <beágyazott SQL utasítás>
2. Változók használata V→:V
3. Vezérlés:

```
                ⎡NOT FOUND   ⎤ ⎡GO TO cimke ⎤
WHENEVER        ⎢SQL WARNING ⎥ ⎢CONTINUE    ⎥
                ⎣SQL ERROR   ⎦ ⎣STOP        ⎦
```

172

173. dia

Beágyazott (embedded) SQL II.

Egyetlen sort eredményező lekérdezések:
A SELECT utasításban rendszerint az
INTO :változónév szerepel

173

174. dia

Beágyazott (embedded) SQL III.

Több sort eredményező lekérdezések:

Egy sormutatót (CURSOR) kell deklarálni, ezzel lehet a sorok között lépegetni (NEXT, PRIOR).

A CURSOR használatával módosítani is lehet a sorokat.

174

175. dia

Beágyazott (embedded) SQL IV.

Dinamikus SQL

A fordításkor még (részben) nem ismert SQL utasítások.

PREPARE

176. dia

4GL program generátorok

1. FORM (űrlap)

2. REPORT (jelentés)

3. MENU

177. dia

Továbbfejlesztett modellek

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

178. dia

Az EER modell

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

A hierarchiában a superclass tulajdonságai öröklődnek (inheritance).

Relációkkal megvalósítható.

178

179. dia

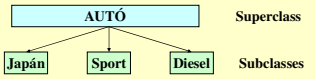
Az EER modell felosztása

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).

179

180. dia

Példa EER modellre



A subclass-ok nem diszjunktak, a felosztás részleges!

180

181. dia

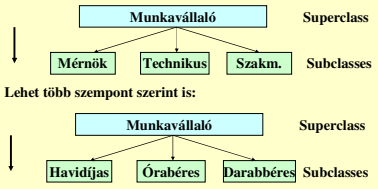
Az EER modell használata

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

181

182. dia

Példa Specializációra

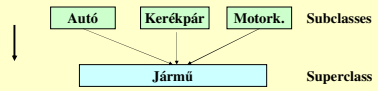


Lehet több szempont szerint is:

182

183. dia

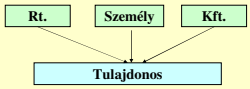
Példa Generalizációra



183

184. dia

Kategória



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

184

185. dia

Nested Relational Model

Nem 1NF, ezért nevezik N1NF modellnek is.

Egy osztály sémája

Oszám	Onév	Ovez	Dolgozó			Ocím
			Dnév	Gyerekek		
				Gynév	Gykor	

185

186. dia

Structural Data Model 1.

Az eredeti relációs modell továbbfejlesztése.

Két típus:
1.) Relations
2.) Connections

186

187. dia

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.

187

188. dia

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.

188

189. dia

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.).

189

190. dia

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.

190

191. dia

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.**

191

192. dia

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.

192

193. dia

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.

193

194. dia

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).

194

195. dia

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
```

195

196. dia

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).**

196

197. dia

Az Objectstore rendszer

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

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

197

198. dia

Az OQL

Object Query Language

Egy objektum orientált nyelv kiterjesztésének fogható fel.
Az SQL funkcióinak megfelelő, de azokkal nem teljesen megegyező utasításokat tartalmaz.
A lényeg az objektum, a reláció fogalma ehhez képest másodlagos.

198

199. dia

Az SQL3

OODB használatára készült szabvány.
A lényeg a reláció, az objektum fogalma ehhez képest másodlagos.

Felűlről kompatibilis a korábbi SQL szabványokkal.

199

200. dia

Új lehetőségek

- **Felhasználó által definiált típusok**
- **Táblák közötti öröklés**
- **Típus konstruktorok**
- **Felhasználó által definiált eljárások és függvények**
- **Nagy objektumok kezelésének támogatása**

200

201. dia

Felhasználó által definiált típusok

- **Abstract Data Type (ADT)**
- **Row Type**
- **Collection Types**

201

202. dia

Abstract Data Type (ADT) I.

Attribútumok és operációk egy entitásban.
Az attribútumok lehetnek „stored” és „virtual” típusúak.
A „stored attribute”egy attribútum név és egy adattípus segítségével adható meg (az adattípus maga is lehet ADT). Ehhez automatikusan tartozik egy get és egy set metódus.
A „virtual attribute”esetén az értéket egy függvény adja meg.

202

203. dia

Abstract Data Type (ADT) II.

Az operációk rutinokként tárolódnak.
Az ADT lehet „subtype”-ja egy „supertype”-nak, mely örökli annak tulajdonságait (Többszörös öröklés is megengedett).
Egy ADT perzisztensen csak táblák oszlopaiként tárolható.

203

204. dia

Row Type

Mezőnév/Adattípus párokból áll.
Egy változóban tárolható, így lehet egy rutin argumentuma, ill. egy függvány által visszaadott érték.
Lehet neve(Named Row Type).

204

205. dia

Collection Types

Halmazok, listák, multihalmazok definiálhatóak.
A táblák oszlopaiban ezek is tárolhatók.

205

206. dia

Táblák közötti öröklés

Egy tábla lehet „subtable”-ja egy „supertable”-nak, mely örökli annak tulajdonságait, de természetesen definiálhatók benne új oszlopok is.
(Független az ADT „subtype” tulajdonságától!)

206

207. dia

Egységbe zárás

Az ADT minden attribútuma és metódusa lehet
Public (azaz minden megfelelő jogosítvánnyal rendelkező felhasználó számára látható),
Private (azaz csak az ADT-n belül látható),
Protected (azaz az ADT-n és annak leszármazottain belül látható).

207

208. dia

Öröklés

Az ADT-nál „supertype” , „subtype” , között.
Tábláknál „supertable” és „subtable” között.

208

209. dia

Többalakúság

Az ADT „subtype”-ban a metódusok újradefiniálhatók (létezik az „overloading).

209

210. dia

Műveletek

Az SQL3-ban a „hagyományos” programozás műveletei is használhatóak:

- **Értékadás**
- **CALL, RETURN**
- **CASE**
- **IF, THEN, ELSE, ELSEIF**
- **LOOP, WHILE REPEAT**

210

211. dia

Deduktív adatbázisok

A logikai programozás eszközeivel dolgozik. Deklaratív nyelvet (pl. PROLOG) használ. ezen írja le a *tényeket* és a *szabályok*at, a már ismert tények és szabályok felhasználásával juthatunk új tényekhez.

211

212. dia

Adatbázis kezelő architektúrák

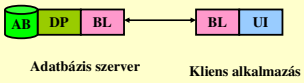
Három fő rész:
1.) Data processing (fizikai adatkezelés).
2.) Business logic (adatvédelem és –integritás, tranzakció kezelés, stb.).
3.) User interface

Hol helyezkednek el?

212

213. dia

Kliens-szerver architektúra

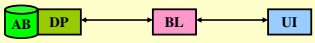


Adatbázis szerver **Kliens alkalmazás**

213

214. dia

Többrétegű architektúra



Adatbázis szerver | Középső réteg (middle-tier) | „Sovány” („Thin”) kliens alkalmazás

214

215. dia

Tárolt rutinok

A szerveren tárolódik és hajtódnak végre.

CREATE PROCEDURE procnév
(IN|OUT | INOUT parnév partípus)
törzs

CREATE FUNCTION funcnév
RETURNS rettípus
IN | OUT | INOUT parnév partípus)
törzs

215

216. dia

Tárolt rutinok hívása

PROCEDURE esetén
CALL procnév(paraméterek)

FUNCTION esetén
kifejezésben használható
(a RETURN utáni érték tér vissza)

217. dia

Utasítások tárolt rutinban 1.

BEGIN … END
több utasítást összefog

DECLARE varnév típus
[DEFAULT érték]
A begin-end között él.

SET varnév=kifejezés

218. dia

Utasítások tárolt rutinban 2.

SELECT oszlopnév INTO varnév táblakif
Csak egy sor!

218

219. dia

Vezérlés tárolt rutinban 1.

IF feltétel THEN utasítások
[ELSEIF feltétel THEN utasítások]
…
[ELSE utasítások]
END IF

220. dia

Vezérlés tárolt rutinban 2.

CASE case_érték
WHEN feltétel THEN utasítások
[WHEN feltétel THEN utasítások]
…
[ELSE utasítások]
END CASE

221. dia

Vezérlés tárolt rutinban 3.

CASE
WHEN feltétel THEN utasítások
[WHEN feltétel THEN utasítások]
…
[ELSE utasítások]
END CASE

222. dia

Vezérlés tárolt rutinban 4.

[kezdőcimke:] LOOP
utasítások
END LOOP [végcimke]

LEAVE cimke

ITERATE címke

223. dia

Vezérlés tárolt rutinban 5.

[kezdőcimke:] REPEAT
utasítások
UNTIL feltétel
END REPEAT [végcimke]

[kezdőcimke:] WHILE feltétel DO
utasítások
END WHILE [végcimke]

224. dia

Kurzorok tárolt rutinban

DECLARE curnév CURSOR FOR select_ut
változó deklaráció után!

OPEN curnév

FETCH curnév INTO
varnév CLOSE curnév

225. dia

Triggerek

CREATE TRIGGER név idő esemény ON táblanév FOR EACH ROW utasítás

Idő: BEFORE | AFTER
Esemény: INSERT | UPDATE | DELETE
Több utasítás BEGIN… END

Változtatásnál OLD.oszlopnév és NEW.oszlopnév használható

DROP TRIGGER tábla.név

226. dia

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.

226

227. dia

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.

227

228. dia

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.

228

229. dia

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.

229

230. dia

Elemzések

1. Forrás nyelő elemzés
A használat gyakorisága, módja.
2. ABC elemzés
A „nélkülözhetetlenség” foka.
3. Érzékenység elemzés
A költség/teljesítmény arány.

230

231. dia

Konzisztencia, konvergencia

Létezik

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

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

231

232. dia

Szinkronizációs protokollok

A. Központosított protokollok

a.) A központi zárellenőrzés

b.) A zseton módszer

c.) Az elsődleges példány módszer

B.) Osztott protokollok

Az időbélyeg módszer

232

233. dia

Adatvédelem

a.) Fizikai védelem

b.) Ügyviteli védelem

c.) Algoritmikus védelem

233

234. dia

Algoritmikus védelem

a.) felhasználó azonosítás, partner azonosítás

b.) hozzáférés védelem

c.) rejtjelezés

d.) üzenethitelesítés

e.) digitális kézjegy

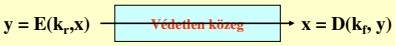
234

235. dia

Rejtjelezés

Legyen „megfejthetetlen”.

Mivel sok kell, legyen:



235

236. dia

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.**

236

237. dia

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)

237

238. dia

A nyilvános (rejtő) kulcsú kódolás

a.) MIT módszer (prímfelbontás).

b.) Merkle-Hellmann módszer (hátizsák probléma).

238

239. dia

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

239

240. dia

Kulcsgenerálás

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

240

241. dia

Kulcskiosztás

Példák:

a.) Alapkulcsok
b.) Merkle „rejtvény” módszere
c.) A "hatványozós" módszer

242. dia

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.

243. dia

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.

244. dia

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).

245. dia

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.

246. dia

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.

247. dia

Felhasználó azonosítás

Személy azonosítás:

a.) jelszóvédelem
b.) fizikai azonosító használata
c.) személyi jellemzők

248. dia

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.

249. dia

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.

250. dia

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.

250

251. dia

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.

251

252. dia

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*.

252

253. dia

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.

253

254. dia

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**

254

255. dia

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)

255

256. dia

Adatfolyamok feldolgozása I.

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

Forgalom figyelés: hálózati, banki, közlek
stb.

Naplózás: meteorológiai, egészségügyi, ipar

256

257. dia

Adatfolyamok feldolgozása II.

Új eszközök (részben az SQL kiterjesztés

Stanford University:
CQL (Continuus Query Language)

Cornell University:
COUGAR

257

258. dia

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.)

258

259. dia

DSS (Decision Support System)

OLAP (On Line Analitical Processing)

Megoldásai:

ROLAP
MOLAP

259

260. dia

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.)**

260

261. dia

ROLAP

Relational On Line Analitical Processing:

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

261

262. dia

MOLAP

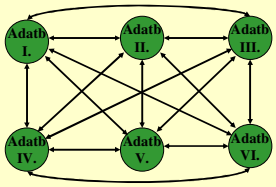
Multidimensional On Line Analitical Processing:

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.

262

263. dia

Adatbázisok összekapcsolása



263

264. dia

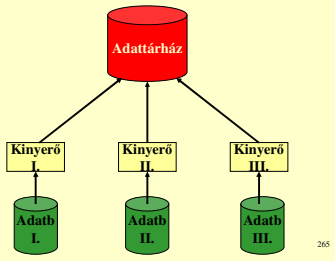
Data Warehouse: Adattárház

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

264

265. dia

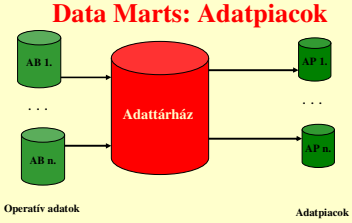
Adattárház



265

266. dia

Data Marts: Adatpiacok



266

267. dia

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.

267

268. dia

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.

268

269. dia

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.

269

270. dia

Tudásfeltárás

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

Több tudományágat átfogó kutatási terület.

Adatbányászat: az adatok összefüggéseinek feltárása.

270

271. dia

Lépések

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

271

272. dia

Minta feladat

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

272

273. dia

Adatkiválasztás

Kiválasztjuk a szükséges adatokat az operatív adatbázisokból.

Pl.
ügyfélszám
név
cím
előfizetési dátum
magazin neve

273

274. dia

Tisztítás

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

274

275. dia

Bővítés

Hozzáveszünk újabb adatokat:
- születési idő
- jövedelem
- van-e autója
- van-e háza
- stb.

275

276. dia

Szűkítés

Kihagyhatunk sorokat:
pl. nincs kitöltve (?)

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

Jól meggondolni, mert elveszíthetünk fontos információkat!

276

277. dia

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!

277

278. dia

Adatbányászat

Sokféle technika lehet, néhány ezek közül:

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

278

279. dia

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!)

279

280. dia

Statisztikai technikák

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

281. dia

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.

282. dia

Vigyázat! II.

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

283. dia

Vizuális technikák

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

283

284. dia

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!

284

285. dia

Megjegyzés

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

285

286. dia

Vigyázat!

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

286

287. dia

Szomszédság

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

287

288. dia

Vigyázat!

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

Túl sok dimenzió esetén nem lesznek jellegzetes osztályok.

288

289. dia

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!

289

290. dia

Vége

290