# Adatbázisok II – Ellenőrző kérdések

**1. Mit hívunk statikus, és mit dinamikus adatbázisnak? (1 pont)**

Statikus adatbázis – ritkán módosul, a lekérdezések gyorsasága a fontosabb

Dinamikus adatbázis – gyakran módosul, ritkán végzünk lekérdezést

**2. Fogalmazzunk meg 3 célt, amire az indexelés kiválasztásánál figyelni kell! (3 pont)**

gyors lekérdezés, gyors adatmódosítás, minél kisebb tárolási terület

**3. Mit tételezünk fel, mivel arányos a beolvasás, kiírás költsége? (1 pont)**

A beolvasás, kiírás költsége arányos a háttértár és a memória között mozgatott blokkok számával.

**4. Adjuk meg az alábbi paraméterek jelentését! l, b, B, T, bf, M, I(A) (7 pont)**

l – rekordméret

b – blokkméret

B – fájl mérete blokkokban

T – rekordok száma

bf – blokkolási faktor (mennyi rekord fér el egy blokkban – b/l alsó egészrésze)

M – memória mérete blokkokban

I(A) – képméret („A” oszlopban szereplő különböző értékek száma)

**5. Adjuk meg RxS méretét blokkokban kifejezve! (2 pont)**

$B(R \times S) = T(S) * B(R) + T(R) * B(S)$

**6. Mit jelent az egyenletességi feltétel? (1 pont)**

Feltesszük, hogy A=a feltételnek eleget tevő rekordokból nagyjából egyforma számú rekord van.

**7. Mekkora adategységet olvas az író-olvasó fej? (1 pont)**

Oracle esetén 8K az alapértelmezés.

**8. Mitől függhet a blokkméret? (1 pont)**

operációs rendszer, hardver, adatbázis-kezelő

**9. Egyenletességi feltétel esetén hány blokkból áll a $\sigma_{A=a}(R)$ lekérdezés eredménye? (1 pont)**

$B(\sigma_{A=a}(R)) = B(R) / I(A)$

**10. Soroljunk fel legalább 7 különböző fájlszervezési módszert? (7 pont)**

kupac, hasító index, rendezett állomány, elsődleges index (ritka index), másodlagos index (sűrű index), többszintű index, $B^+$-fa, $B^*$-fa

**11. Kupac szervezés esetén mennyi a keresés költsége legrosszabb esetben? (1 pont)**

B (a fájl mérete blokkokban)

**12. Kupac szervezés esetén mennyi a beszúrás költsége? (1 pont)**

utolsó blokkba tesszük a rekordot: 1 olvasás + 1 írás

**13. Mit mond meg a h(x) hasító függvény értéke? (1 pont)**

A h(x) hasító függvény mondja meg, hogy melyik kosárba tartozik a rekord, ha x volt az indexmező értéke a rekordban.

**14. Mikor jó egy hasító függvény és ilyenkor milyen hosszúak a blokkláncok? (2 pont)**

Akkor jó egy hasító függvény, ha nagyjából egyforma blokkláncok keletkeznek, ekkor egy blokklánc B/K blokkból áll.

**15. Mennyi a $\sigma_{A=a}(R)$ lekérdezés keresési költsége jó hasító index esetén? (1 pont)**

B/K legrosszabb esetben

**16. Ha túl nagynak választjuk a K-t hasításkor, akkor ez milyen problémát okozhat? (1 pont)**

Sok olyan blokklánc keletkezhet, amely 1 blokkból fog állni, és a blokkban is csak 1 rekord lesz. (B helyett T számú blokkban tároljuk az adatokat)

**17. Milyen keresésre nem jó a hasító indexelés? (1 pont)**

Intervallumos keresésre

**18. Mit jelent a dinamikus hasító indexelés és milyen két fajtáját ismerjük? (3 pont)**

Előre nem rögzítjük a kosarak számát, a kosarak száma beszúráskor, törléskor változhat.

Fajtái: kiterjeszthető, lineáris

**19. Kiterjeszthető hasítás esetén a h(K) érték alapján melyik kosárba kerül a rekord? (2 pont)**

A h(k) k hosszú kódjának vegyük az i hosszú elejét, és azt a kosarat, amelynek kódja a h(k) kezdő szelete. Ha van hely a kosárban, tegyük bele, ha nincs, akkor nyissunk egy új kosarat, és a következő bit alapján osszuk ketté a telített kosár rekordjait. Ha ez a bit megegyezik, vegyük a következőt.

**20. Milyen probléma keletkezhet kiterjeszthető hasító index esetén és mi rá a megoldás? (2 pont)**

Probléma: Ha az új sorok hasító értékének eleje sok bitben megegyezik, akkor hosszú ágak keletkezhetnek (nincs kiegyensúlyozva a fa).

Megoldás: A bináris gráfot teljessé tesszük. A gráfot tömbbel ábrázolhatjuk. Ekkor minden kosár azonos szintes lesz, de közös blokkjai is lehetnek a kosaraknak. Túlcsordulás esetén a kosarak száma duplázódik.

**21. Lineáris hasító index esetén mikor nyitunk meg új kosarat? (1 pont)**

Akkor nyitunk meg új kosarat, ha egy előre megadott értéket elér a kosarakra jutó átlagos rekordszám.

**22. Lineáris hasító index esetén a h(K) érték alapján melyik kosárba kerül a rekord? (2 pont)**

Ha n kosarunk van, akkor a hasító függvény értékének utolsó log(n) bitjével megegyező sorszámú kosárba tesszük, ha van benne hely. Ha nincs, akkor hozzáláncolunk egy új blokkot és abba tesszük. Ha nincs megfelelő sorszámú kosár, akkor abba a sorszámú kosárba tesszük, amely csak az első bitjében különbözik a keresett sorszámtól.

**23. Rendezett állomány esetén adjuk meg a bináris (logaritmikus) keresés lépéseit! (4 pont)**

1, Beolvassuk a középső blokkot

2, Ha nincs benne az A = a értékű rekord, akkor eldöntjük, hogy a blokklánc második felében, vagy az első felében szerepelhet-e egyáltalán

3, Beolvassuk a felezett blokklánc középső blokkját

4, Addig folytatjuk, amíg megtaláljuk a rekordot, vagy a vizsgálandó maradék blokklánc már csak 1 blokkból áll

**24. Mennyi a keresési költség rendezett mező esetében? (1 pont)**

$\log_2(B)$

**25. Mennyi a keresési költség rendezett mező esetében, ha gyűjtő blokkokat is használunk? (1 pont)**

$\log_2(B - G) + G$

**26. Mennyi a keresési költség rendezett mező esetében, ha minden blokkot félig üresen hagyunk? (1 pont)**

$\log_2(2 * B)$

**27. Milyen mindig az indexrekord szerkezete? (1 pont)**

(a, p) – ahol „a” egy érték az indexelt oszlopban, „p” pedig egy blokkmutató, amely arra a blokkra mutat, amelyben az A=a értékű rekordot tároljuk

**28. Adjuk meg az elsődleges index 5 jellemzőjét! (5 pont)**

- főfájl is rendezett

- csak 1 elsődleges indexet lehet megadni (mert csak egyik mező szerint lehet rendezett a főfájl)

- elég a főfájl minden blokkjának legkisebb rekordjához készíteni indexrekordot

- indexrekordok száma: T(I) = B

- indexrekordból sokkal több fér egy blokkba, mint a főfájl rekordjaiból

**29. Mit hívunk fedőértéknek? (1 pont)**

A legnagyobb olyan indexértéket, amely a keresett értéknél kisebb vagy egyenlő.

**30. Mennyi a keresési költség elsődleges index esetén? (1 pont)**

$\log_2(B(I))$

**31. Adjuk meg a másodlagos index 5 jellemzőjét! (5 pont)**

- főfájl rendezetlen

- több másodlagos indexet is meg lehet adni

- a főfájl minden rekordjához kell készíteni indexrekordot

- indexrekordok száma: T(I) = T

- indexrekordból sokkal több fér egy blokkba, mint a főfájl rekordjaiból

**32. Hogyan keresünk a másodlagos indexben és mennyi a keresés költsége? (5 pont)**

- az indexben keresés az index rendezettsége miatt bináris kereséssel történik: $\log_2(B(I))$

- a talált indexrekordban szereplő blokkmutatónak megfelelő blokkot még be kell olvasni: $1 + \log_2(B(I))$

- az elsődleges indexnél rosszabb a keresési idő, mert több az indexrekord

**33. Mit hívunk klaszterszervezésű táblának? (1 pont)**

Klaszterszervezés egy tábla esetén egy „A” oszlopra:

- az azonos „A” értékű sorok fizikailag egymás után blokkokban helyezkednek el

- az első találat után az összes találatot megkapjuk soros beolvasással

**34. Mit hívunk klaszterindexnek? (1 pont)**

Klaszterszervezésű fájl esetén egy indexet az A oszlopra.

**35. Mikor mondjuk, hogy 2 tábla klaszterszervezésű? (1 pont)**

- a közös oszlopokon egyező sorok egy blokkban, vagy fizikailag egymás utáni blokkokban helyezkednek el

- összekapcsolás esetén az összetartozó sorokat soros beolvasással megkaphatjuk

**36. Ha t szintű indexet használunk, mennyi a keresési költség blokkműveletek számában mérve? (1 pont)**

$\log_2(B(I^{(t)})) + t$ blokkolvasás

**37. Ha t szintű indexet használunk, a legfelső szinten milyen keresést használunk? (1 pont)**

bináris keresést

**38. Ha t szintű indexet használunk és a legfelső szint 1 blokkból áll, akkor mennyi a keresési költség? (1 pont)**

t+1 blokkolvasás

**39. Ha t szintű indexet használunk, mennyi az indexszintek blokkolási faktora és miért? (2 pont)**

Minden szint blokkolási faktora megegyezik, mert egyforma hosszúak az indexrekordok.

**40. Ha t szintű indexet használunk, vezessük le, hogy hány blokkból áll a legfelső szint! (12 pont)**

| | FŐFÁJL | 1. szint | 2. szint | ... | t. szint |
|---|---|---|---|---|---|
| blokkok száma | B | B/bf(I) | $B/bf(I)^2$ | ... | $B/bf(I)^t$ |
| rekordok száma | T | B | B/bf(I) | ... | $B/bf(I)^{(t-1)}$ |
| blokkolási faktor | bf | bf(I) | bf(I) | ... | bf(I) |

**41. Ha t szintű indexet használunk, és a legfelső szint 1 blokkból áll, abból milyen egyenlet következik és mi a megoldása t-re? (2 pont)**

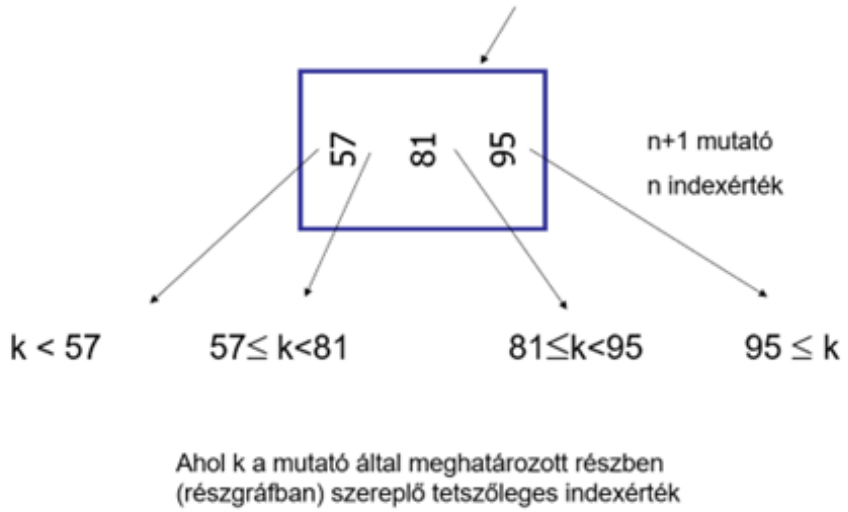
$B / bf(I)^t = 1$, azaz: $t = \log_{bf(I)}(B)$

**42. Mi a két legfontosabb jellemzője a $B^+$-faindexnek? (2 pont)**

$B^+$ fa: minden blokk legalább 50%-ban telített (a szerkezeten kívül a telítettséget biztosító karbantartó algoritmusokat is beleértjük)

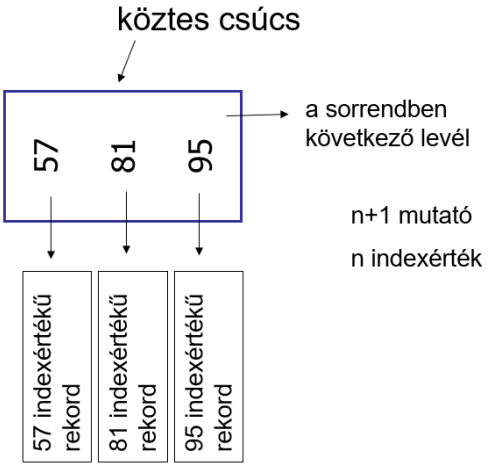
$B^*$ fa: minden blokk legalább 66%-ban telített

**43. Egy példa alapján szemléltessük a köztes csúcs jellemzőit $B^+$-fa index esetén! (8 pont)**

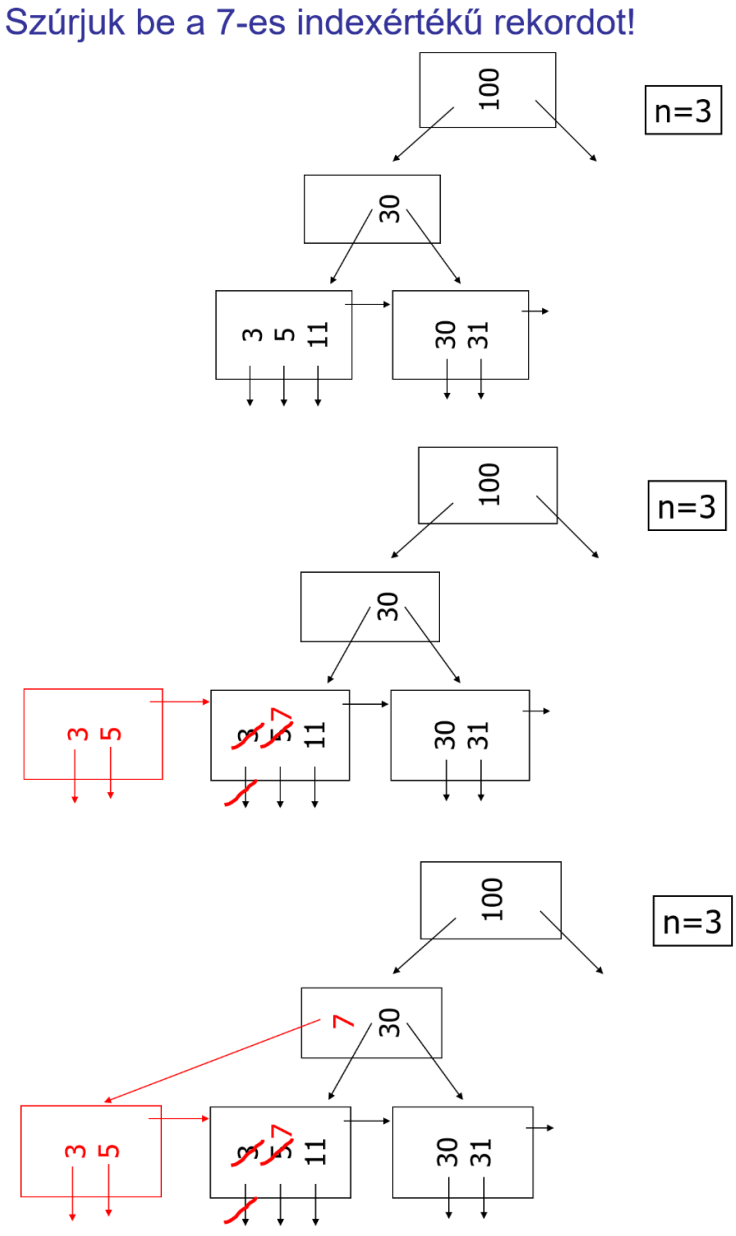



Ahol k a mutató által meghatározott részben (részgráfban) szereplő tetszőleges indexérték

**44. Egy példa alapján szemléltessük a levél csúcs jellemzőit B⁺-fa index esetén! (5 pont)**

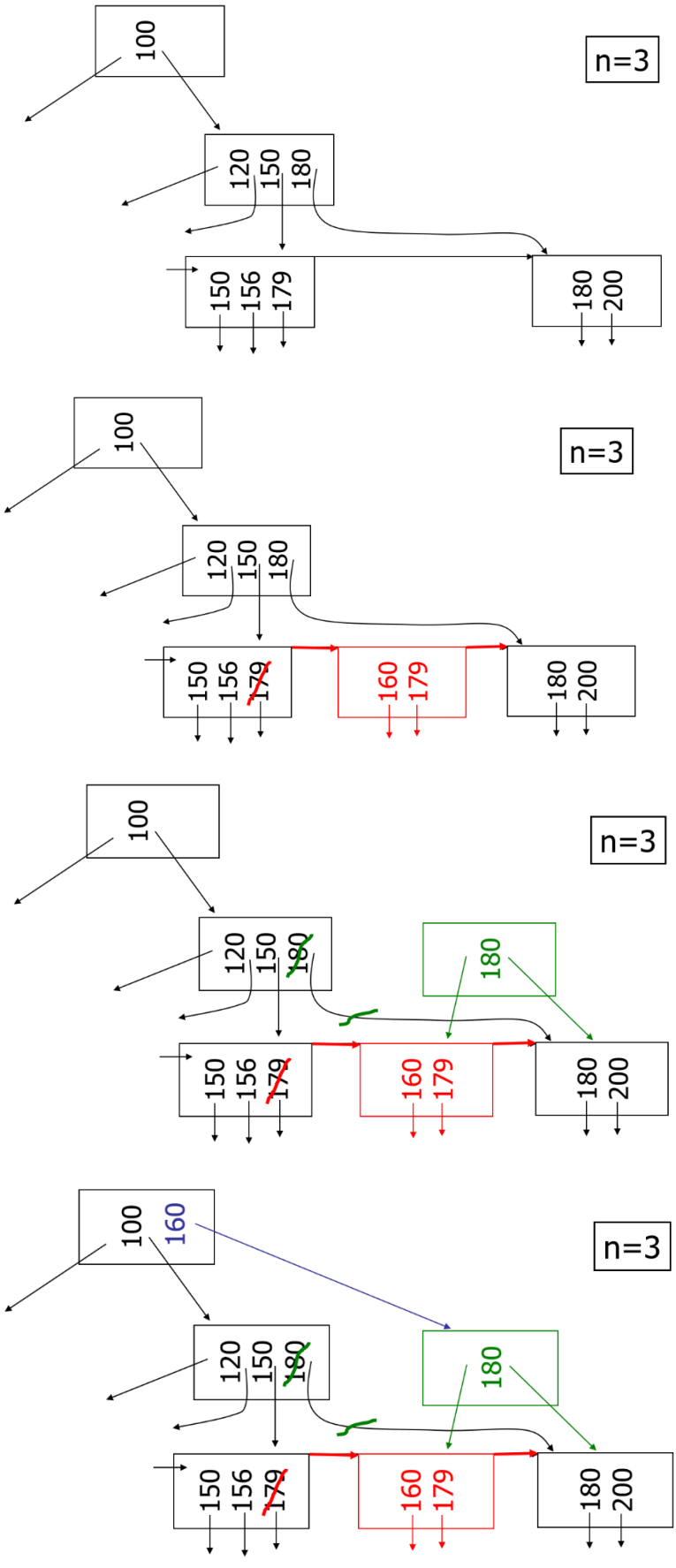



**45. Mutassunk példát, mikor beszúráskor egy levélcsúcsot kettéosztunk B⁺-fa index esetén! (5 pont)**

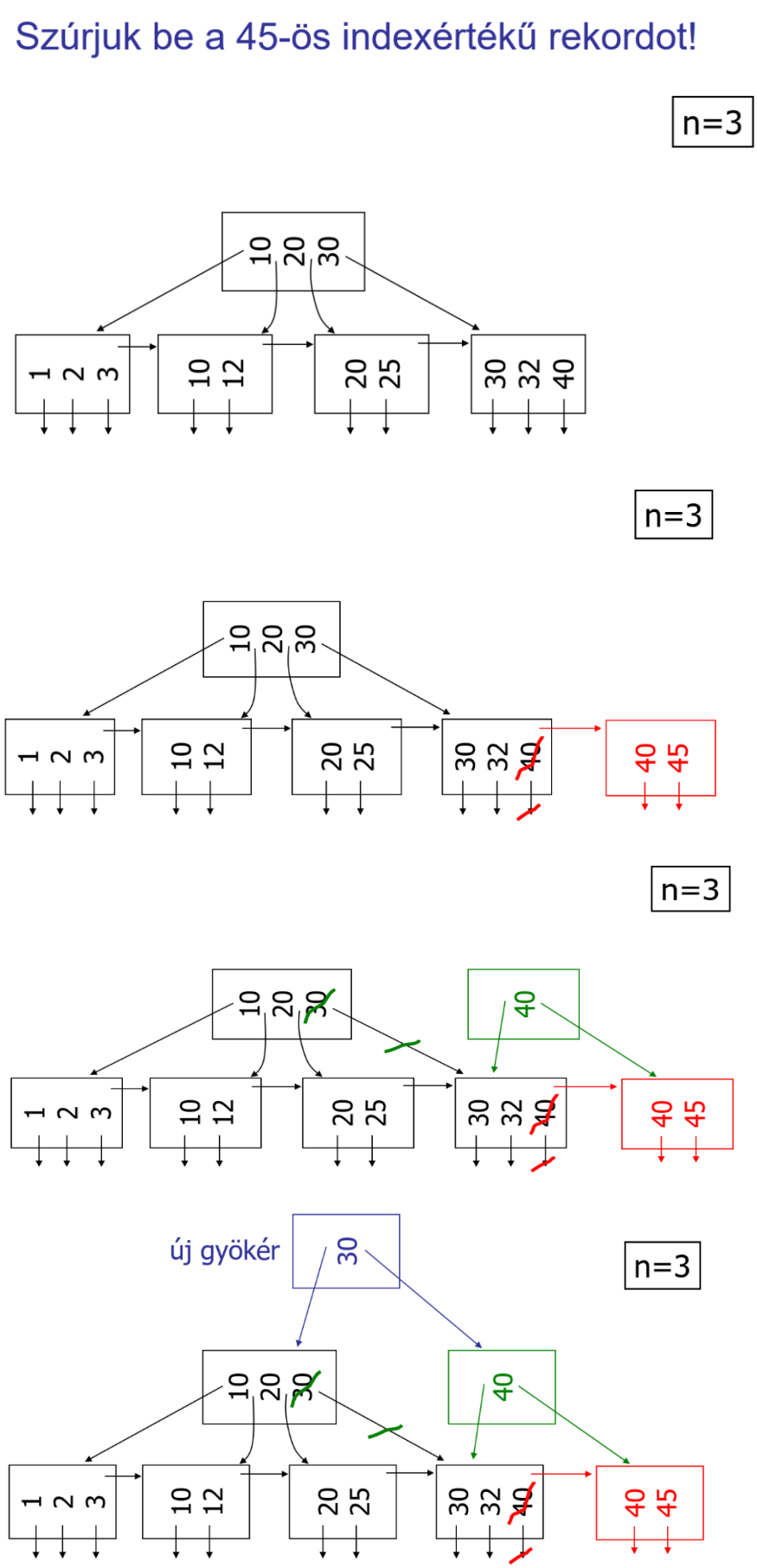


**46. Mutassunk példát, mikor beszúráskor egy köztes csúcsot kettéosztunk $B^+$-fa index esetén! (5 pont)**

Szúrjuk be a 160-as indexértékű rekordot!

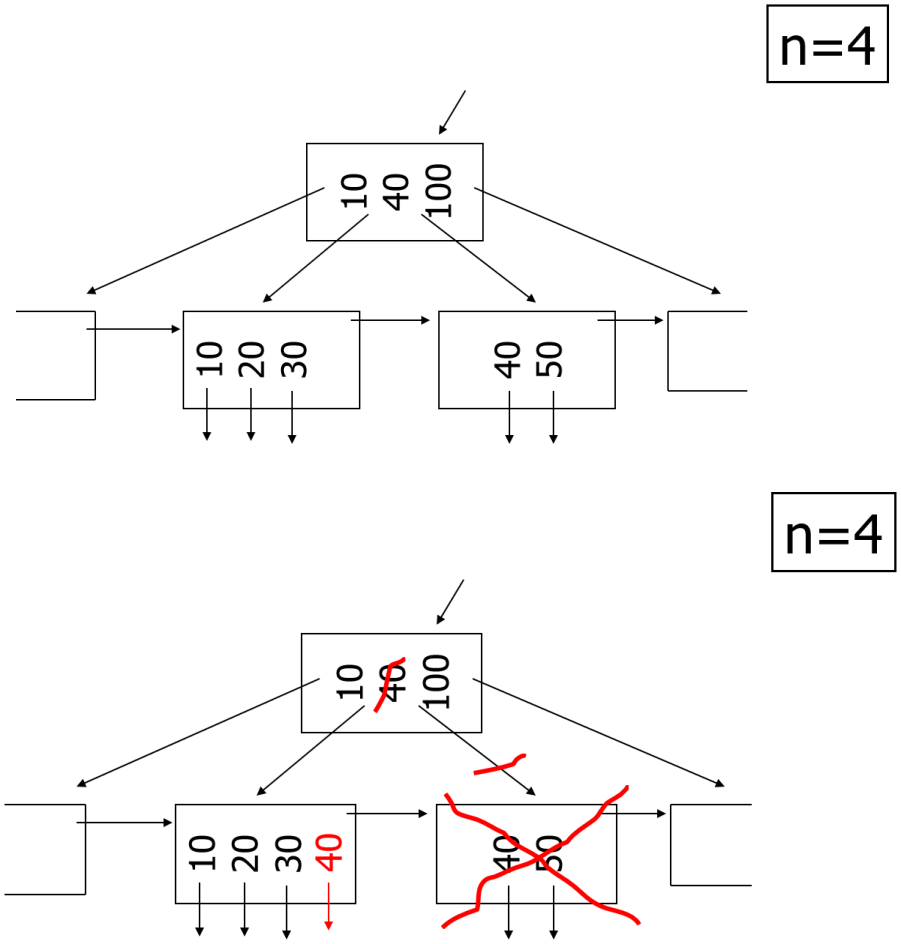


**47. Mutassunk példát, mikor beszúráskor nő a $B^+$-fa index magassága! (5 pont)**



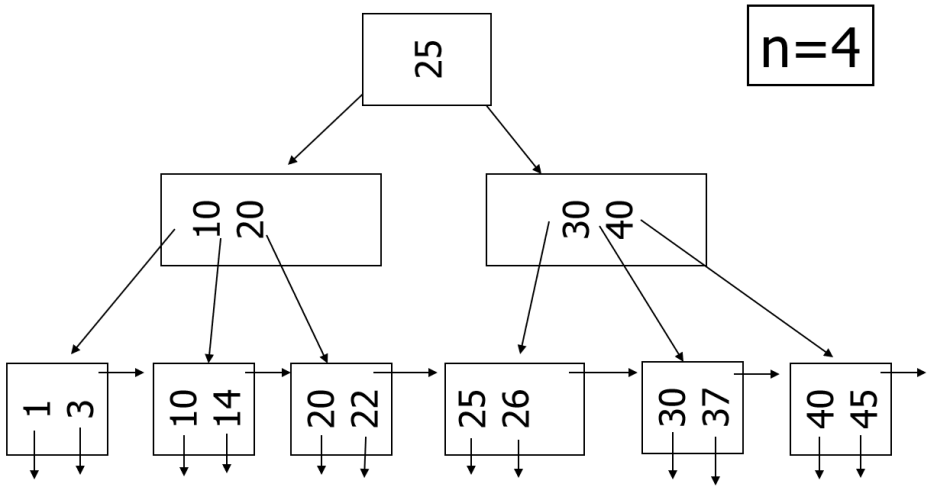
**48. Mutassunk példát, mikor törléskor megszüntetünk egy levélcsúcsot $B^+$-fa index esetén! (5 pont)**

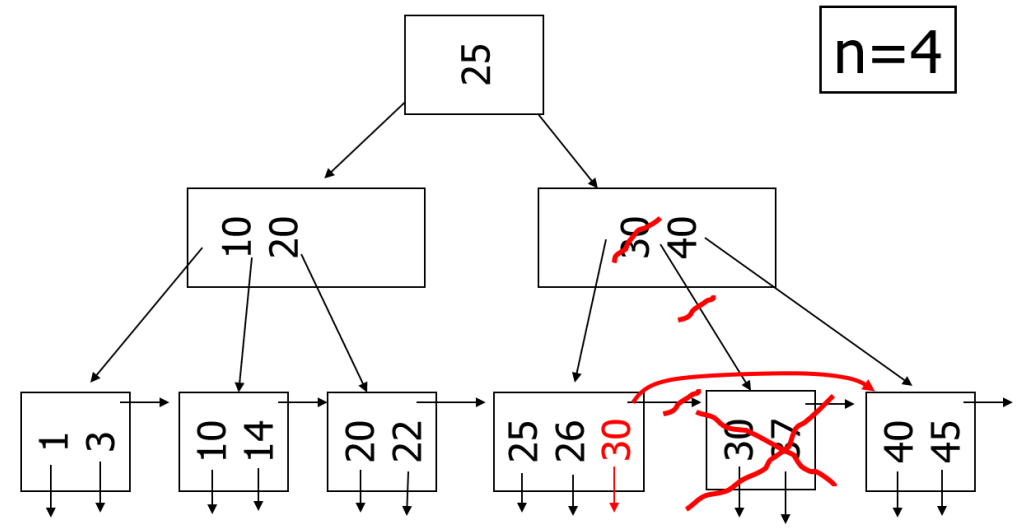
Töröljük az 50-es indexértékű rekordot!

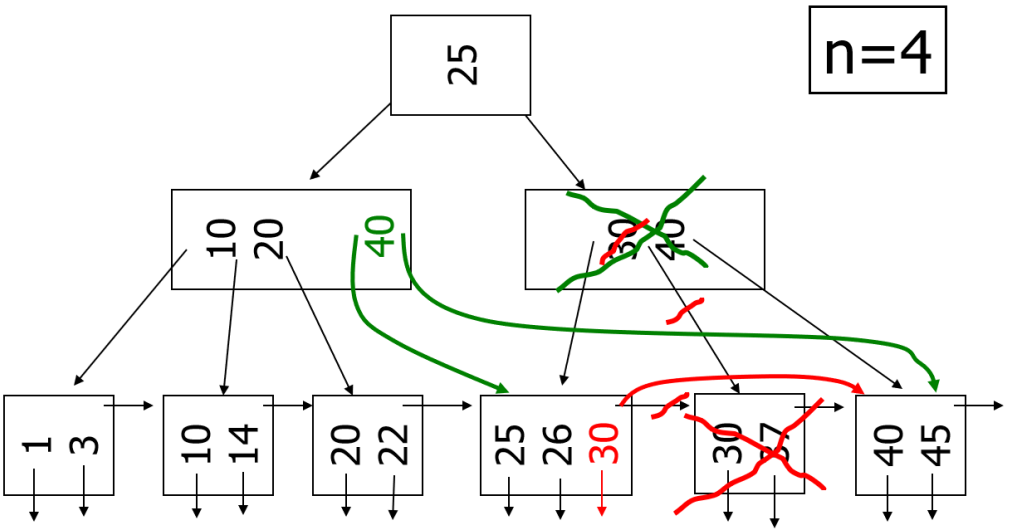


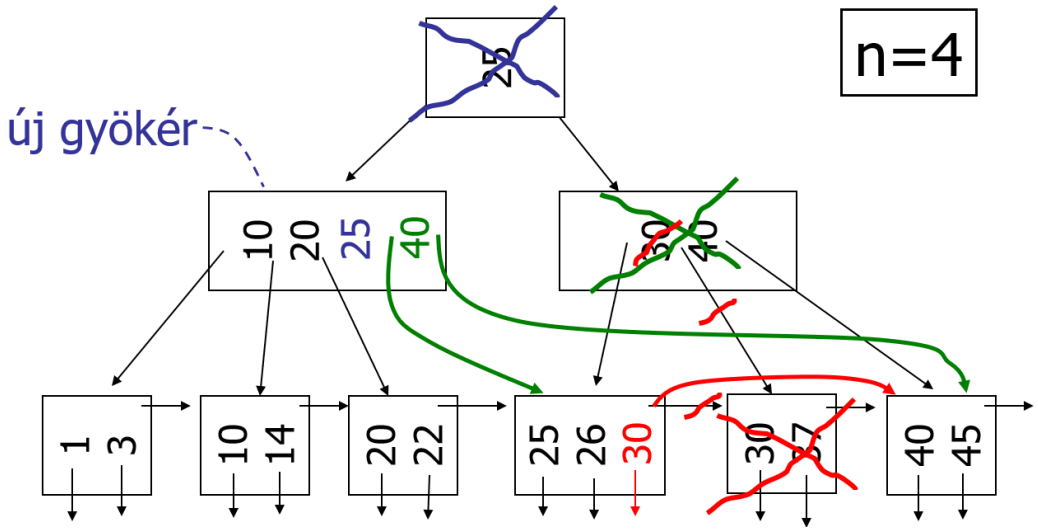

**49. Mutassunk példát, mikor törléskor csökken a $B^+$-fa index magassága! (5 pont)**

Töröljük a 37-es indexértékű rekordot!




n=4
25
10 20
30 40
1 3
10 14
20 22
25 26 30
30 37
40 45


n=4
25
10 20
40
30 40
1 3
10 14
20 22
25 26 30
30 37
40 45


n=4
új gyökér
25
10 20 25 40
30 40
1 3
10 14
20 22
25 26 30
30 37
40 45

**50. Mutassunk példát arra, mikor egy kevés elemszámú oszlopra bitmap indexet készítünk! (2 pont)**

| CUSTOMER # | MARITAL_ STATUS | REGION | GENDER | INCOME_ LEVEL |
|---|---|---|---|---|
| 101 | single | east | male | bracket_1 |
| 102 | married | central | female | bracket_4 |
| 103 | married | west | female | bracket_2 |
| 104 | divorced | west | male | bracket_4 |
| 105 | single | central | female | bracket_2 |
| 106 | married | central | female | bracket_3 |

| REGION='east' | REGION='central' | REGION='west' |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 0 | 1 | 0 |

**51. Mutassunk példát arra, mikor logikai feltételek kiértékelését bitmap vektorműveletekre vezetjük vissza! (7 pont)**

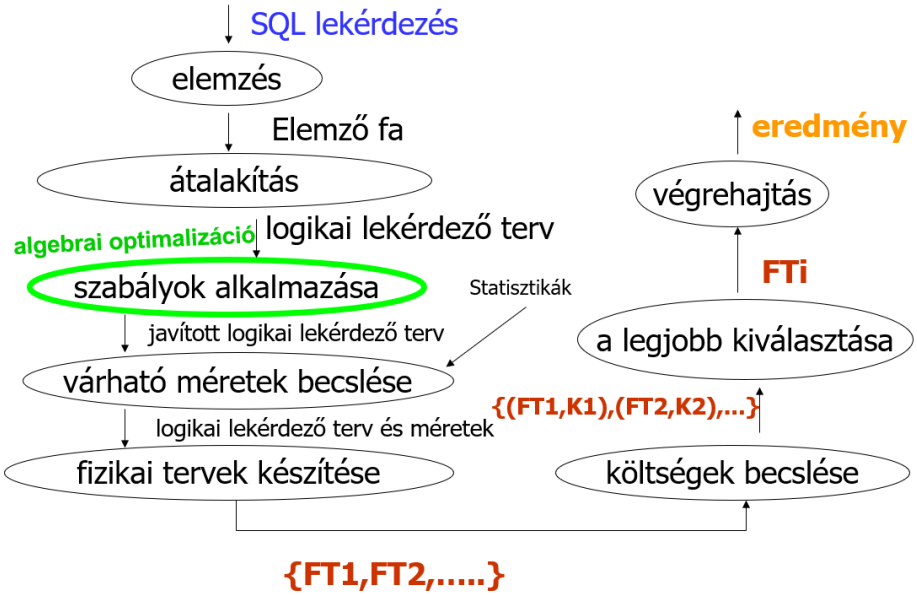
```
SELECT COUNT(*)
FROM CUSTOMER
WHERE MARITAL_STATUS = 'married' AND REGION IN ('central','west');
```

| status = 'married' | | region = 'central' | | region = 'west' | | | | |
|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | | 0 | | 0 | | 0 |
| 1 | | 1 | | 0 | | 1 | | 1 |
| 1 | AND | 0 | OR | 1 | = | 1 | AND | 1 | = | 1 |
| 0 | | 0 | | 1 | | 0 | | 1 | | 0 |
| 0 | | 1 | | 0 | | 0 | | 1 | | 0 |
| 1 | | 1 | | 0 | | 1 | | 1 | | 1 |

**52. Mi a lekérdezések optimalizálásának a célja és miket használunk fel ehhez? (5 pont)**

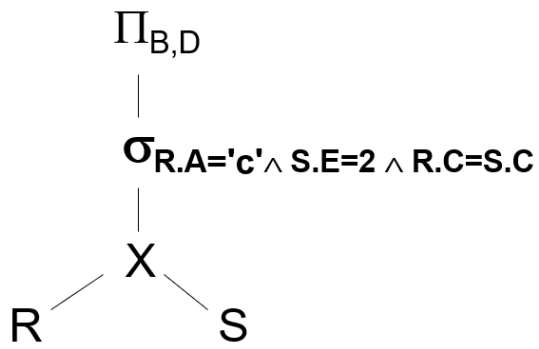
A lekérdezéseket gyorsabbá akarjuk tenni a táblára vonatkozó paraméterek, indexek, statisztikák ismeretében és általános érvényű tulajdonságok, heurisztikák segítségével.

**53. Adjuk meg a lekérdezések optimalizálásának folyamatábráját! (19 pont)**




**54. Adjuk meg egy egyszerű relációs algebrai kifejezést és gráfos ábrázolását! (4 pont)**

$\Pi_{B,D}$ [ $\sigma_{R.A='c' \wedge S.E=2 \wedge R.C = S.C}$ (RXS)]




**55. Milyen költségmodellt használunk relációs algebrai optimalizálás esetében? (2 pont)**

A kiszámítás költsége arányos a relációs algebrai kifejezés részkifejezéseinek megfelelő relációk tárolási méreteinek összegével.

**56. Mi a módszer lényege relációs algebrai optimalizálás esetében? (3 pont)**

A műveleti tulajdonságokon alapuló ekvivalens átalakításokat alkalmazunk, hogy várhatóan kisebb méretű relációk keletkezzenek.

**57. Miért mondjuk, hogy az eljárás heurisztikus relációs algebrai optimalizálás esetén? (2 pont)**

Azért, mert nem az argumentum relációk valódi méretével számol.

**58. Miért nem egyértelmű az eredmény relációs algebrai optimalizálás esetén? (4 pont)**

Az átalakítások sorrendje nem determinisztikus, így más sorrendben végrehajtva az átalakításokat más végeredményt kaphatunk, de mindegyik általában jobb költségű, mint amiből kiindultunk.

**59. A relációs algebrai kifejezésfában melyek az unáris csúcsok? (3 pont)**

Olyan csúcsok, amelyeknek egy gyereke van: $\sigma,\Pi,\rho$

**60. A relációs algebrai kifejezésfában melyek a bináris csúcsok? (3 pont)**

Olyan csúcsok, amelyeknek két gyereke van: $-,\cup,\times$

**61. A relációs algebrai kifejezésfában mik a levélcsúcsok? (2 pont)**

konstans relációk vagy relációs változók

**62. Mit hívunk ekvivalens relációs algebrai kifejezéseknek? (3 pont)**

E1(r1, …, rk) és E2(r1, …, rk) relációs algebrai kifejezések ekvivalensek (E1 ≅ E2), ha tetszőleges r1, …, rk relációkat véve E1(r1, …, rk) = E2(r1, …, rk).

**63. Hány szabálycsoportot adunk meg relációs algebrai optimalizáláskor és mi jellemző ezekre? (4 pont)**

11 szabályt adunk meg. A szabályok olyan állítások, amelyek kifejezések ekvivalenciáját fogalmazzák meg. Bizonyításuk könnyen végiggondolható. Az állítások egy részében a kifejezések szintaktikus helyessége egyben elégséges feltétele is az ekvivalenciának.

**64. Adjuk meg a relációs algebrai optimalizálás kommutatív szabályait! (3 pont)**

$E1 \times E2 \cong E2 \times E1$

$E1 \ |x|\ E2 \cong E2 \ |x|\ E1$

$E1 \ |x|_{\Theta}\ E2 \cong E2 \ |x|_{\Theta}\ E1$

**65. Adjuk meg a relációs algebrai optimalizálás asszociatív szabályait! (3 pont)**

$(E1 \times E2) \times E3 \cong E1 \times (E2 \times E3)$

$(E1 \ |x|\ E2) \ |x|\ E3 \cong E1 \ |x|\ (E2 \ |x|\ E3)$

$(E1 \ |x|_{\Theta}\ E2) \ |x|_{\Theta}\ E3 \cong E1 \ |x|_{\Theta}\ (E2 \ |x|_{\Theta}\ E3)$

**66. Adjuk meg a vetítésre vonatkozó összevonási, bővítés szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen $\underline{A}$ és $\underline{B}$ két részhalmaza E reláció oszlopainak úgy, hogy $\underline{A} \subseteq \underline{B}$.

Ekkor: $\Pi_{\underline{A}}(\Pi_{\underline{B}}(E)) \cong \Pi_{\underline{A}}(E)$

**67. Adjuk meg a kiválasztások felcserélhetőségére, felbontására vonatkozó szabályt relációs algebrai optimalizálás esetén! (3 pont)**

Legyen F1 és F2 az E reláció oszlopain értelmezett kiválasztási feltétel.

Ekkor: $\sigma_{F1 \wedge F2}(E) \cong \sigma_{F1}(\sigma_{F2}(E)) \cong \sigma_{F2}(\sigma_{F1}(E))$

**68. Adjuk meg a kiválasztás és vetítés felcserélhetőségére vonatkozó szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen F az E relációnak csak az $\underline{A}$ oszlopain értelmezett kiválasztási feltétel.

Ekkor: $\Pi_{\underline{A}}(\sigma_F(E)) \cong \sigma_F(\Pi_{\underline{A}}(E))$

**69. Adjuk meg a kiválasztás és vetítés felcserélhetőségére vonatkozó általánosított szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen F az E relációnak csak az $\underline{A} \cup \underline{B}$ oszlopain értelmezett kiválasztási feltétel, ahol $\underline{A} \cap \underline{B} = \emptyset$.

Ekkor: $\Pi_{\underline{A}}(\sigma_F(E)) \cong \Pi_{\underline{A}}(\sigma_F(\Pi_{\underline{A} \cup \underline{B}}(E)))$

**70. Adjuk meg a kiválasztás és szorzás felcserélhetőségére vonatkozó szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen F az E1 reláció oszlopainak egy részhalmazán értelmezett kiválasztási feltétel.

Ekkor: $\sigma_F(E1 \times E2) \cong \sigma_F(E1) \times E2$

**71. Adjuk meg a kiválasztás és szorzás felcserélhetőségére vonatkozó speciális szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen i = 1,2 esetén Fi az Ei reláció oszlopainak egy részhalmazán értelmezett kiválasztási feltétel, legyen továbbá F = F1 $\wedge$ F2.

Ekkor: $\sigma_F(E1 \times E2) \cong \sigma_{F1}(E1) \times \sigma_{F2}(E2)$

**72. Adjuk meg a kiválasztás és szorzás felcserélhetőségére vonatkozó általánosított szabályt relációs algebrai optimalizálás esetén! (3 pont)**

Legyen F1 az E1 reláció oszlopainak egy részhalmazán értelmezett kiválasztási feltétel, legyen F2 az E1 x E2 reláció oszlopainak egy részhalmazán értelmezett kiválasztási feltétel úgy, hogy mindkét sémából legalább egy oszlop szerepel benne, legyen továbbá F = F1 $\wedge$ F2.

Ekkor: $\sigma_F(E1 \times E2) \cong \sigma_{F2}(\sigma_{F1}(E1) \times E2)$

**73. Adjuk meg a kiválasztás és egyesítés felcserélhetőségére vonatkozó szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen E1, E2 relációk sémája megegyező, és F a közös sémán értelmezett kiválasztási feltétel.

Ekkor: $\sigma_F(E1 \cup E2) \cong \sigma_F(E1) \cup \sigma_F(E2)$

**74. Adjuk meg a kiválasztás és természetes összekapcsolás felcserélhetőségére vonatkozó szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen F az E1 és E2 közös oszlopainak egy részhalmazán értelmezett kiválasztási feltétel.

Ekkor: $\sigma_F(E1 |\times| E2) \cong \sigma_F(E1) \ |x|\ \sigma_F(E2)$

**75. Adjuk meg a vetítés és szorzás felcserélhetőségére vonatkozó szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen i=1,2 esetén $\underline{Ai}$ az Ei reláció oszlopainak egy halmaza, valamint legyen $\underline{A} = \underline{A1} \cup \underline{A2}$.

Ekkor: $\Pi_{\underline{A}}(E1 \times E2) \cong \Pi_{\underline{A1}}(E1) \times \Pi_{\underline{A2}}(E2)$

**76. Adjuk meg a vetítés és egyesítés felcserélhetőségére vonatkozó szabályt relációs algebrai optimalizálás esetén! (2 pont)**

Legyen E1 és E2 relációk sémája megegyező, és legyen $\underline{A}$ a sémában szereplő oszlopok egy részhalmaza.

Ekkor: $\Pi_{\underline{A}}(E1 \cup E2) \cong \Pi_{\underline{A}}(E1) \cup \Pi_{\underline{A}}(E2)$

**77. Mutassunk példát, hogy a kivonás és a vetítés nem felcserélhető! (2 pont)**

Legyenek:

E1:

| A | B |
|---|---|
| 0 | 0 |
| 0 | 1 |

E2:

| A | B |
|---|---|
| 0 | 0 |

Ekkor: míg $\Pi_A(E1) - \Pi_A(E2) = \varnothing$

$\Pi_A(E1 - E2)$:

| A |
|---|
| 0 |

**78. Fogalmazzuk meg a relációs algebrai optimalizálás 4 heurisztikus elvét! (4 pont)**

- minél hamarabb szelektáljunk

- próbáljunk természetes összekapcsolásokat képezni

- vonjuk össze az egymás utáni unáris műveleteket

- keressünk közös részkifejezéseket

**79. Miért érdemes hamarabb szelektálni relációs algebrai optimalizálás esetén? (1 pont)**

A részkifejezések így várhatóan kisebb relációk lesznek.

**80. Miért érdemes természetes összekapcsolásokat képezni szorzások helyett relációs algebrai optimalizálás esetén? (1 pont)**

Az összekapcsolás így hatékonyabban kiszámítható, mint a szorzatokból történő kiválasztás.

**81. Miért érdemes az unáris műveleteket összevonni relációs algebrai optimalizálás esetén? (1 pont)**

Csökken tőle a műveletek száma, és általában a kiválasztás kisebb relációt eredményez, mint a vetítés.

**82. Miért érdemes a közös részkifejezéseket megkeresni relációs algebrai optimalizálás esetén? (1 pont)**

A közös részkifejezéseket elég csak egyszer kiszámolni a kifejezés kiértékelése során.

**83. A relációs algebrai optimalizálás algoritmusának mi az inputja és mi az outputja? (2 pont)**

INPUT: relációs algebrai kifejezés kifejezésfája

OUTPUT: optimalizált kifejezésfa optimalizált kiértékelése

**84. Mi a relációs algebrai optimalizálás algoritmusának 1. lépése (az alkalmazott szabályok felsorolása nélkül)? (2 pont)**

A kiválasztásokat bontsuk fel.

**85. Mi a relációs algebrai optimalizálás algoritmusának 2. lépése (az alkalmazott szabályok felsorolása nélkül)? (2 pont)**

A kiválasztásokat vigyük olyan mélyre a kifejezésfában, amilyen mélyre csak lehet.

**86. Mi a relációs algebrai optimalizálás algoritmusának 3. lépése (az alkalmazott szabályok felsorolása nélkül)? (2 pont)**

A vetítéseket vigyük olyan mélyre a kifejezésfában, amilyen mélyre csak lehet.

**87. Mi a relációs algebrai optimalizálás algoritmusának 4. lépése (az alkalmazott szabályok felsorolása nélkül)? (2 pont)**

Ha egy relációs változóra vagy konstans relációra közvetlenül egymás után kiválasztásokat vagy vetítéseket alkalmazunk, akkor ezeket vonjuk össze egy kiválasztássá vagy egy vetítéssé, vagy egy kiválasztás utáni vetítéssé, ha lehet. Ezzel megkapjuk az optimalizált kifejezésfát.

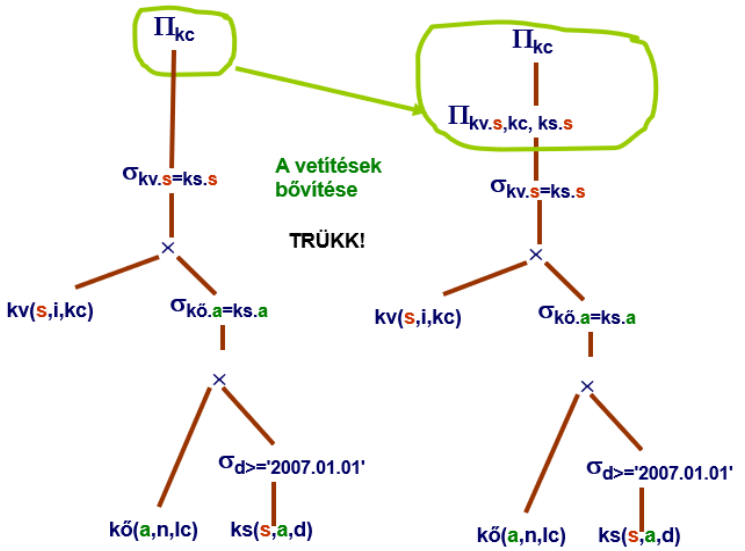
**88. Mi a relációs algebrai optimalizálás algoritmusának 5. lépése (az alkalmazott szabályok felsorolása nélkül)? (4 pont)**

A gráfot a bináris műveletek alapján bontsuk részgráfokra. Minden részgráf egy bináris műveletnek feleljen meg. A részgráf csúcsai legyenek a bináris műveletnek megfelelő csúcs és a csúcs felett a következő bináris műveletig szereplő kiválasztások és vetítések. Ha a bináris művelet a szorzás és a részgráf equi-joinnak felel meg, és a szorzás valamelyik ága nem tartalmaz bináris műveletet, akkor ezt az ágat is vegyük hozzá a részgráfhoz.

**89. Mi a relációs algebrai optimalizálás algoritmusának 6. lépése (az alkalmazott szabályok felsorolása nélkül)? (2 pont)**

Az előző lépésben kapott részgráfok is fát képeznek. Az optimális kiértékeléshez ezt a fát értékeljük ki alulról felfelé haladva, tetszőleges sorrendben.

**90. Adjunk meg egy példát, amiben a vetítések bővítése trükköt alkalmazzuk és indokoljuk, hogy mire jó ez! (8 pont)**




Indoklás: jobb oldalon felvettünk egy olyan vetítést, ami az alatta szereplő kiválasztással felcserélhető, ezután pedig még lejjebb tudjuk vinni ezeket a vetítéseket.

**91. Mennyi az SC(A,R) szelektivitás értéke, ha A kulcs? (1 pont)**

S(A,R) = 1

**92. Mennyi az SC(A,R) szelektivitás értéke, ha A nem kulcs (a jelölések magyarázatát is adjuk meg)? (1 pont)**

$S(A,R) = N_R / V(A,R)$

$N_R$ – rekordok száma R-ben

V(A,R) – A attribútum egyedi értékeinek száma R-ben

**93. Mennyi rendezett táblában a bináris keresés átlagos költsége, ha minden találatot be kell olvasni (a jelölések magyarázatát is adjuk meg)? (3 pont)**

$|\log_2 B_R| + m$, ahol: $m = ceil(SC(A,R)/F_R) - 1$

$B_R$ – R reláció lapjainak száma

m – további lapok, amelyeket be kell olvasni

SC(A,R) – A kiválasztási számossága R-ben

$F_R$ – blokkolási faktor

**94. Mennyi $B^+$-fa típusú elsődleges index esetén az átlagos keresési költség, ha minden találatot be kell olvasni (a jelölések magyarázatát is adjuk meg)? (2 pont)**

$HT_I + ceil(SC(A,R)/F_R)$

$HT_I$ – I index szintjeinek száma

SC(A,R) – A kiválasztási számossága R-ben

$F_R$ – blokkolási factor

**95. Mennyi $B^+$-fa típusú másodlagos index esetén az átlagos keresési költség, ha minden találatot be kell olvasni (a jelölések magyarázatát is adjuk meg)? (2 pont)**

$HT_I + SC(A,R)$

$HT_I$ – I index szintjeinek száma

SC(A,R) – A kiválasztási számossága R-ben

**96. A $\sigma_{\theta_1 \wedge \theta_2 \ldots \wedge \theta_n}$ lekérdezésnek adjuk meg kétféle kiszámítási módját! (6 pont)**

1, Egyszerű kiválasztást végzünk a legkisebb költségű $\theta_i$-re:

– a fennmaradó θ feltételek szerint szűrjük az eredményt

2, Több index:

- válasszuk ki a $\theta_i$-khez tartozó indexeket, keressünk az indexekben és adjuk vissza a RID-ket, az eredmény a RID-k metszete

**97. A $\sigma_{\theta_1 \vee \theta_2 \ldots \vee \theta_n}$ lekérdezésnek adjuk meg kétféle kiszámítási módját! (3 pont)**

- több index – RID-k uniója

- lineáris keresés

**98. Milyen adatbázis műveletekhez kell rendezés? (5 pont)**

SELECT DISTINCT esetén:

- π-hez szükséges a duplikált értékek kiszűrése

- rendezés

Halmazműveletek duplikált értékeinek kiszűrése esetén:

- $R \cap S$

- $R \cup S$

- rendezés

**99. Milyen két fajtája van a rendezésnek? (2 pont)**

- belső rendezés (ha a rekordok befférnek a memóriába)

- külső rendezés

**100. Külső összefésülő rendezésnél mire jó a rendező lépés? (1 pont)**

rendezett futamokat hoz létre

**101. Külső összefésülő rendezésnél mire jó az összevonási lépés? (1 pont)**

rendezett futamokat fésül össze

**102. Külső összefésülő rendezésnél mikor kell több menetben végezni az összevonási lépést? (2 pont)**

Akkor, ha N > M.

N – lefoglalt lapok száma

M – beolvasott lapok száma

**103. Külső összefésülő rendezésnél mennyi a rendező lépés költsége? (2 pont)**

$2 * B_R$

$B_R$ – R reláció lapjainak száma

**104. Külső összefésülő rendezésnél mennyi összevonandó futam van kezdetben? (2 pont)**

$$\left\lceil \frac{B_R}{M} \right\rceil$$

$B_R$ – R reláció lapjainak száma

M – beolvasott lapok száma R relációból a memóriába

**105. Külső összefésülő rendezésnél mennyi az összes menetek száma? (2 pont)**

$$\left\lceil \log_{M-1}\left(\frac{B_R}{M}\right) \right\rceil$$

$B_R$ – R reláció lapjainak száma

M – beolvasott lapok száma R relációból a memóriába

**106. Külső összefésülő rendezésnél mennyi blokkot olvasunk minden menetben? (2 pont)**

$2 * B_R$

$B_R$ – R reláció lapjainak száma

**107. Külső összefésülő rendezésnél mennyi a teljes költség, a végeredmény kiírása nélkül? (4 pont)**

$2 * B_R + 2 * B_R * \left\lceil \log_{M-1}\left(\frac{B_R}{M}\right) \right\rceil - B_R$

$B_R$ – R reláció lapjainak száma

M – beolvasott lapok száma R relációból a memóriába

**108. A vetítés milyen három lépés megvalósításából áll? (3 pont)**

kezdeti átnézés + rendezés + végső átnézés

**109. Soroljuk fel az összekapcsolás 5 megvalósítását! (5 pont)**

skatulyázott ciklusos (nested loop)-, blokk-skatulyázott ciklusos (block-nested loop)-, indexelt skatulyázott ciklusos-, összefésüléses rendező-, hasításos összekapcsolás

**110. Skatulyázott (NestedLoop) összekapcsolásnál mennyi a költség legjobb esetben? (3 pont)**

A legjobb eset akkor következik be, ha a kisebb reláció elfér a memóriában, ekkor ezt használjuk belső relációnak.

Költség: $B_R + B_S$

**111. Skatulyázott (NestedLoop) összekapcsolásnál mennyi a költség legrosszabb esetben? (3 pont)**

A legrosszabb eset akkor következik be, ha mindkét relációból csak 1-1 lap fér bele a memóriába, ilyenkor S-t minden R-beli rekordnál végig kell olvasni.

Költség: $N_R * B_S + B_R$

**112. Blokk-Skatulyázott (BlockNestedLoop) összekapcsolásnál mennyi a költség legjobb esetben? (3 pont)**

A legjobb eset akkor következik be, ha a kisebb reláció elfér a memóriában, ekkor ezt használjuk belső relációnak.

Költség: $B_R + B_S$

**113. Blokk-Skatulyázott (BlockNestedLoop) összekapcsolásnál mennyi a költség legrosszabb esetben? (3 pont)**

A legrosszabb eset akkor következik be, ha mindkét relációból csak 1-1 lap fér bele a memóriába, ilyenkor S-t minden R-beli lapnál végig kell olvasni.

Költség: $B_R * B_S + B_R$

**114. Indexelt összekapcsolásnál mennyi a költség? (3 pont)**

$B_R + N_R * c$

c – a belső relációból index szerinti kiválasztás költsége

A kevesebb rekordot tartalmazó reláció legyen a külső.

**115. Rendezéses-Összefésüléses összekapcsolásnál mennyi a költség? (3 pont)**

rendezés költsége + $B_R + B_S$

**116. Hasításos összekapcsolásnál mennyi a költség? (3 pont)**

$2*(B_R+B_S) + (B_R+B_S)$

**117. Hasításos összekapcsolásnál mekkora méretű kosarakat képezünk? (2 pont)**

Alkalmazzuk $h_1$-et az összekapcsolási mezőre és felosztjuk a rekordokat a memóriában elférő részekre.

**118. Hány sora van a $\sigma_{A=v}(R)$ lekérdezés eredményének? (2 pont)**

SC(A,R)

**119. Hány sora van a $\sigma_{A\leq v}(R)$ lekérdezés eredményének? (2 pont)**

$$N_R * \frac{v - \min(A,R)}{\max(A,R) - \min(A,R)}$$

**120. Hány sora van a $\sigma_{\theta_1 \wedge \theta_2 \wedge \ldots \wedge \theta_n}(R)$ lekérdezés eredményének? (2 pont)**

$$N_R * [(s_1/N_R) * (s_2/N_R) * \ldots (s_n/N_R)]$$

**121. Hány sora van a $\sigma_{\theta_1 \vee \theta_2 v \ldots \vee \theta_n}(R)$ lekérdezés eredményének? (2 pont)**

$$N_R * (1 - [(1 - s_1/N_R) * (1 - s_2/N_R) * \ldots * (1 - s_n/N_R)])$$

**122. Hány sora van az R x S lekérdezés eredményének? (2 pont)**

$N_R * N_S$

**123. Hány sora van az R ⋈ S lekérdezés eredményének, ha R ∩ S = ∅? (2 pont)**

$N_R * N_S$

**124. Hány sora van az R ⋈ S lekérdezés eredményének, ha R ∩ S kulcs R-en? (2 pont)**

legfeljebb $N_S$

**125. Hány sora van az R ⋈ S lekérdezés eredményének, ha R ∩ S idegen kulcs R-hez? (2 pont)**

$N_S$

**126. Hány sora van az R ⋈ S lekérdezés eredményének, ha R ∩ S = {A} sem R-nek, sem S-nek nem kulcsa? (2 pont)**

$N_R * N_S / V(A,S)$ vagy $N_S * N_R / V(A,R)$

**127. Mi a szabályos zárójelezések számának rekurzív képlete? (2 pont)**

$T(1) = 1$

$T(n) = \Sigma T(i)T(n-i)$

**128. Mennyi n tagú Join fa van? (2 pont)**

T(n)*n!, ahol T(n) az n elem szabályos zárójelezéseinek száma

**129. 5 tagú összekapcsolás sorrendjének legjobb tervét dinamikus programozási elvet alkalmazva hogyan számoljuk ki? (3 pont)**

BestPlan(A,B,C,D,E) = min of (
BestPlan(A,B,C,D) ⋈E,
BestPlan(A,B,C,E) ⋈D,
BestPlan(A,B,D,E) ⋈C,
BestPlan(A,C,D,E) ⋈B,
BestPlan(B,C,D,E) ⋈A )

**130. Több-tagú összekapcsolás szuboptimális sorrendjét milyen algoritmussal lehet előállítani, és a tartalmazási hálón milyen irányban halad a kiértékelés? (2 pont)**

Selinger-algoritmussal, a kiértékelés pedig alulról felfelé halad

**131. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek melyik három kiértékelését hasonlítottuk össze, és melyik volt a legjobb ezek közül? (4 pont)**

balról jobbra, balról jobbra és a memóriában összekapcsolva a harmadik táblával, a középső ténytábla soraihoz kapcsolva a szélső dimenziótáblákat – az utolsó módszer a leghatékonyabb

**132. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek három kiértékelésénél milyen indexeket tételeztünk fel? (2 pont)**

Q.B-re és S.C-re klaszterindexünk van

**133. Az R(A,B) JOIN S(B,C) lekérdezés eredményében mennyi a sorok száma? (2 pont)**

$T_R = T_S = T$

**134. Az R(A,B) JOIN S(B,C) lekérdezés eredménye hány blokkból áll? (2 pont)**

$B_R = B_S = B$

**135. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek balról jobbra (a) kiértékelésénél milyen költségek összege lesz a teljes költség, és mennyi a teljes költség? (5 pont)**

A teljes JOIN I/O költsége:

- Az 1. join költsége: $B+T*B/I$
- Az 1. join kiírása: $2*T*B/I$
- A 2. join költsége: $2*T*B/I+[(T^2/I*B)]/I$
- A teljes output kiírása: $3*T^2*B/I^2$

Teljes költség: $B+5*T*B/I+4*T^2*B/I^2$

**136. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek balról jobbra (b) kiértékelésénél mennyit lehet megspórolni és mennyi a teljes költség? (5 pont)**

Meg tudjuk spórolni: $2*(2*T*B/I)$

Teljes költség: $B+T*B/I+4*T^2*B/I^2$

**137. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek c) kiértékelésénél (középső ténytáblához indexek alapján kapcsoljuk a dimenziótáblákat) milyen költségek összege lesz a teljes költség, és mennyi a teljes költség? (4 pont)**

- Q beolvasása: B
- Q és S olvasása R minden sorára: $T*(B/I+B/I)$
- A teljes output kiírása: $3*T^2*B/I^2$

Teljes költség: $B+2*T*B/I+3*T^2*B/I^2$

**138. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek c) és b) kiértékelésének költségei hogy aránylanak egymáshoz, és milyen feltétel szükséges ehhez? (2 pont)**

Nagyméretű táblák esetén a T/l hányados nagy szám lesz, ezért a négyzetes tag jóval nagyobb lesz, mint a lineáris tag, vagyis a „középső ténytábla soraihoz kapcsolva a szélső dimenziókat” módszer lesz a leghatékonyabb. Ha harmadik és a második módszer arányát tekintjük, akkor azt mondhatjuk, hogy ez az arány ¾-hez tart, ha T/l tart a végtelenbe. Vagyis, ha T/l elég nagy, akkor a harmadik módszer költsége nagyjából ¾-e a második módszernek.

**139. A legjobb átfutás mit optimalizál? (2 pont)**

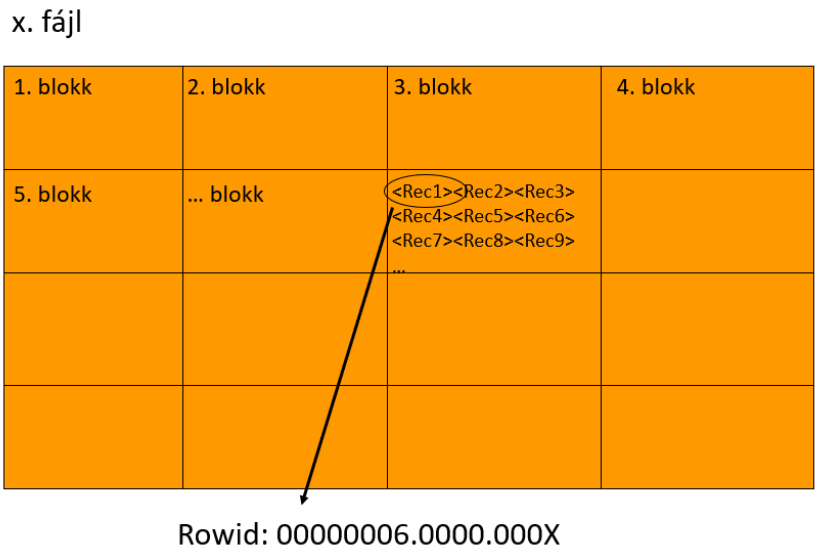
Minden sort minél hamarabb. Először számol, aztán gyorsan visszatér.

**140. A legjobb válaszidő mit optimalizál? (2 pont)**

Első sort minél hamarabb. Számítás közben már visszatér (ha lehetséges).

**141. Adjuk meg a ROWID szerkezetét, és egy példát is rá Oracle esetében! (2 pont)**

ROWID: <Blokk>.<Sor>.<Fájl>




**142. Mi az "Explain plan for<SQL-utasítás>" utasítás hatása? (2 pont)**

Elmenti a tervet (sorforrások + műveletek) a Plan_Table-be.

**143. Jellemezzük a SELECT * FROM emp WHERE rowid= '00004F2A.00A2.000C' utasítást! (4 pont)**

- egy sort keresünk meg
- azonnal a blokkra meg és kiszűri a sort
- a leggyorsabb módszer egy sor kinyerésére (ha tudjuk a rowid-t)

**144. Mit jelent a konzisztens állapot és mit jelent a konzisztens adatbázis? (2 pont)**

A konzisztens állapot kielégíti az összes előre megszabott feltételt (megszorítást). A konzisztens adatbázis konzisztens állapotú adatbázis.

**145. Mit hívunk tranzakciónak és mi jellemző rá? (4 pont)**

Tranzakció: Konzisztenciát megtartó adatkezelő műveletek sorozata.

Jellemzői:

Ha T tranzakció konzisztens állapotból indul, és T csak egyedül futna le, akkor T konzisztens állapotban hagyja az adatbázist.

**146. Mit jelent a tranzakció atomossági tulajdonsága? (2 pont)**

A tranzakció „mindent vagy semmit" jellegű végrehajtása (vagy teljesen végrehajtjuk, vagy egyáltalán nem hajtjuk végre).

**147. Mit jelent a tranzakció konzisztencia tulajdonsága? (2 pont)**

Az a feltétel, hogy a tranzakció végrehajtása után is teljesüljenek az adatbázisban előírt konzisztenciamegszorítások (integritási megszorítások), azaz az adatelemekre és a közöttük levő kapcsolatokra vonatkozó elvárások.

**148. Mit jelent a tranzakció elkülönítési tulajdonsága? (2 pont)**

Minden tranzakciónak látszólag úgy kell lefutnia, mintha ez alatt az idő alatt semmilyen másik tranzakciót nem hajtanánk végre.

**149. Mit jelent a tranzakció tartóssági tulajdonsága? (2 pont)**
Ha egyszer egy tranzakció befejeződött, akkor már soha többé nem veszhet el a tranzakciónk az adatbázisok kifejtett hatása.

**150. A tranzakció-feldolgozónak milyen három feladata van? (3 pont)**

naplózás, konkurenciavezérlés, holtpont feloldása

**151. A tranzakciók melyik tulajdonságát biztosítja a naplózás? (1 pont)**

tartósságot

**152. A tranzakciók melyik tulajdonságát biztosítja a konkurenciakezelés? (1 pont)**

A tranzakcióknak úgy kell látszódniuk, mintha egymástól függetlenül, elkülönítve végeznénk őket.

**153. Mi az ütemező feladata? (2 pont)**

Összetett tranzakciók résztevékenységeinek egy olyan sorrendját határozza meg, amely biztosítja azt, hogy ha ebben a sorrenden hajtjuk végre a tranzakciók elemi tevékenyégeit, akkor az összhatás megegyezik azzal, mintha a tranzakciókat egyenként és egységes egészként hajtottuk volna végre.

**154. Mitől sérülhet a konzisztencia? (4 pont)**

tranzakcióhiba, adatbázis-kezelési hiba, hardverhiba, adatmegosztásból származó hiba

**155. A belső társérülés elleni védekezés milyen két lépésből áll? (4 pont)**

1. lépés: Felkészülés a hibára → naplózás

2. lépés: Hiba utáni helyreállítás → a napló segítségével egy konzisztens állapot helyreállítása

**156. Mit hívunk adatbáziselemnek? (2 pont)**

Az adatbáziselem a fizikai adatbázisban tárolt adatok egyfajta funkcionális egyége, amelyeknek értékét tranzakciókkal lehet elérni (kiolvasni) vagy módosítani (kiírni).

**157. A tranzakció és az adatbázis kölcsönhatásának milyen három fontos helyszíne van? (3 pont)**

1, Az adatbázis elemeit tartalmazó lemezblokkok területe

2, A pufferkezelő által használt virtuális vagy valós memóriaterület

3, A tranzakció memóriaterülete

**158. Mit jelent az INPUT(X) művelet? (2 pont)**

Az X adatbáziselemet tartalmazó lemezblokk másolás a memóriapufferbe.

**159. Mit jelent a READ(X,t) művelet? (4 pont)**

Az X adatbáziselem bemásolása a tranzakció t lokális változójába. Ha az X adatbáziselemet tartalmazó blokk nincs a memóriapufferben, akkor előbb végrehajtódik INPUT(X). Ezután kapja meg a t lokáli változó X értékét.

**160. Mit jelent a Write(X,t) művelet? (4 pont)**

A t lokális változó tartalma az X adatbáziselem memóriapufferbeli tartalmába másolódik. Ha az X adatbáziselemet tartalmazó blokk nincs a memóriapufferben, akkor előbb végrehajtódik INPUT(X). Ezután másolódik át a t lokális változó értéke a pufferbeli X-be.

**161. Mit jelent az Output(X) művelet? (2 pont)**

Az X adatbáziselemet tartalmazó puffer kimásolása lemezre.

**162. Adjuk meg az Undo naplózás U1 és U2 szabályát! (4 pont)**

U1: Ha T tranzakció módosítja az X adatbáziselemet, akkor a (T, X régi érték) naplóbejegyzést azelőtt kell a lemezre írni, mielőtt az X új értékét a lemezre írná a rendszer.

U2: Ha a tranzakció hibamentesen befejeződött, akkor a COMMIT naplóbejegyzést csak azután szabad a lemezre írni, ha a tranzakció által módosított összes adatbáziselem már a lemezre íródott, de ezután rögtön.

**163. Adjunk meg egy példát Undo naplózás esetén a lemezre írás sorrendjére! (6 pont)**

| *Lépés* | *Tevékenység* | *t* | *M-A* | *M-B* | *D-A* | *D-B* | *Napló* |
|---|---|---|---|---|---|---|---|
| 1) | | | | | | | <T,START> |
| 2) | READ(A,t) | 8 | 8 | | 8 | 8 | |
| 3) | t := t*2 | 16 | 8 | | 8 | 8 | |
| 4) | WRITE(A,t) | 16 | 16 | | 8 | 8 | <T,A,8> |
| 5) | READ(B,t) | 8 | 16 | 8 | 8 | 8 | |
| 6) | t := t*2 | 16 | 16 | 8 | 8 | 8 | |
| 7) | WRITE(B,t) | 16 | 16 | 16 | 8 | 8 | <T,B,8> |
| 8) | FLUSH LOG | | | | | | |
| 9) | OUTPUT(A) | 16 | 16 | 16 | 16 | 8 | |
| 10) | OUTPUT(B) | 16 | 16 | 16 | 16 | 16 | |
| 11) | | | | | | | <T,COMMIT> |
| 12) | FLUSH LOG | | | | | | |

**164. Adjuk meg Undo naplózás esetén a helyreállítás algoritmusát! (8 pont)**

**(1) Let S = set of transactions with**
**<Ti, start> in log, but no**
**<Ti, commit> (or <Ti, abort>) record in log**
**(2) For each <Ti, X, v> in log,**
**in reverse order (latest → earliest) do:**
**- if Ti ∈ S then { - write (X, v)**
**- output (X)**
**(3) For each Ti ∈ S do**
**- write <Ti, abort> to log**
**(4) Flush log**

**165. Adjunk meg a működés közbeni ellenőrzőpont képzésének lépéseit Undo naplózás esetén! (6 pont)**

1. <START CKPT(T1,...,Tk)> naplóbejegyzés készítése, majd lemezre írása (FLUSH LOG), ahol T1,...,Tk az éppen aktív tranzakciók nevei.
2. Meg kell várni a T1,...,Tk tranzakciók mindegyikének normális vagy abnormális befejeződését, nem tiltva közben újabb tranzakciók indítását.
3. Ha a T1,...,Tk tranzakciók mindegyike befejeződött, akkor <END CKPT> naplóbejegyzés elkészítése, majd lemezre írása (FLUSH LOG).

**166. Ha UNDO naplózás utáni helyreállításkor előbb <ENDCKPT> naplóbejegyzéssel találkozunk, akkor meddig kell visszamenni a napló olvasásában? (2 pont)**

Ekkor tudjuk, hogy az összes még be nem fejezett tranzakcióra vonatkozó naplóbejegyzést a legközelebbi korábbi <START CKPT(T1, …,TK)> naplóbejegyzésig megtaláljuk. Ott viszont megállhatunk, az annál korábbiakat akár el is dobhatjuk.

**167. Ha UNDO naplózás utáni helyreállításkor előbb <STARTCKPT(T1,…,Tk)>naplóbejegyzéssel találkozunk, akkor meddig kell visszamenni a napló olvasásában? (2 pont)**

Ekkor a be nem fejezett tranzakciók közül a legkorábban (t) kezdődött tranzakció indulásáig kell a naplóban visszakeresnünk, annál korábbra nem.

**168. Adjuk meg a REDO naplózás esetén a lemezre írás sorrendjét 5 lépésben! (5 pont)**

1, Ha egy tranzakció v-re módosítja egy X adatbáziselem értékét, akkor egy <T,X,v> bejegyzést kell a naplóba írni.

2, Az adatbáziselemek módosítását leíró naplóbejegyzések lemezre írása.

3, A COMMIT naplóbejegyzés lemezre írása.

4, Az adatbáziselemek értékének cseréje a lemezen.

5, A <T,end>-t bejegyezzük a naplóba, majd kiírjuk lemezre a naplót.

**169. Adjuk meg a REDO naplózás esetén az R1 szabályt! (2 pont)**

Mielőtt az adatbázis bármely X elemét a lemezen módosítanánk, az X módosítására vonatkozó összes naplóbejegyzésnek, azaz <T,X,v>-nek és <T,COMMIT>-nak a lemezre kell kerülnie.

**170. Adjunk meg egy példát REDO naplózás esetén a lemezre írás sorrendjére! (6 pont)**

| *Lépés* | *Tevékenység* | *t* | *M-A* | *M-B* | *D-A* | *D-B* | *Napló* |
|---|---|---|---|---|---|---|---|
| 1) | | | | | | | <T,START> |
| 2) | READ(A,t) | 8 | 8 | | 8 | 8 | |
| 3) | t := t*2 | 16 | 8 | | 8 | 8 | |
| 4) | WRITE(A,t) | 16 | 16 | | 8 | 8 | <T,A,16> |
| 5) | READ(B,t) | 8 | 16 | 8 | 8 | 8 | |
| 6) | t := t*2 | 16 | 16 | 8 | 8 | 8 | |
| 7) | WRITE(B,t) | 16 | 16 | 16 | 8 | 8 | <T,B,16> |
| 8) | | | | | | | <T,COMMIT> |
| 9) | FLUSH LOG | | | | | | |
| 10) | OUTPUT(A) | 16 | 16 | 16 | 16 | 8 | |
| 11) | OUTPUT(B) | 16 | 16 | 16 | 16 | 16 | |
| 12) | | | | | | | <T,END> |
| 13) | FLUSH LOG | | | | | | |

**171. Adjunk meg REDO naplózás esetén a helyreállítás algoritmusát! (8 pont)**

(1) Let S = set of transactions with
<Ti, commit> (and no <Ti, end>) in log
(2) For each <Ti, X, v> in log, in forward
order (earliest → latest) do:
- if Ti ∈ S then { Write(X, v)
Output(X) }
(3) For each Ti ∈ S, write <Ti, end>

**172. Mi jellemző a módosított REDO naplózásra? (8 pont)**

Nem használunk <Ti,end> bejegyzést a befejezett tranzakciókra, helyette a be nem fejezetteket jelöljük meg <Ti,abort>-tal.

**173. Fogalmazzunk meg 3 különbséget az UNDO és REDO naplózás esetén! (3 pont)**

REDO különbsége az UNDO-hoz képest:

1, Az adat változás utáni értékét jegyezzük fel a naplóba

2, Máshová rakjuk a COMMIT-ot

3, Az UNDO protokoll esetleg túl gyakran akar írni → itt el lehet halasztani az írást

**174. Mit hívunk piszkos puffereknek? (1 pont)**

A piszkos pufferek végrehajtott, de lemezre még ki nem írt módosításokat tárolnak.

**175. Adjuk meg a működés közbeni ellenőrzőpont képzésének lépéseit REDO naplózás esetén! (6 pont)**

1. <START CKPT(T1,…,Tk)> naplóbejegyzés elkészítése és lemezre írása, ahol T1,…,Tk az összes éppen aktív tranzakció.
2. Az összes olyan adatbáziselem kiírása lemezre, melyeket olyan tranzakciók írtak pufferekbe, melyek a START CKPT naplóba írásakor már befejeződtek, de puffereik lemezre még nem kerültek.
3. <END CKPT> bejegyzés naplóba írása, és a napló lemezre írása.

**176. Adjuk meg az UNDO/REDO naplózás esetén az UR1 szabályt! (2 pont)**

Mielőtt az adatbázis bármely X elemének értékét – valamely T tranzakció által végzett módosítás miatt – a lemezen módosítanánk, ezt megelőzően a <T,X,v,w> naplóbejegyzésnek lemezre kell kerülnie.

**177. Adjuk meg az UNDO/REDO naplózás esetén a WAL elvet! (2 pont)**

Write After Log elv: előbb naplózunk, utána módosítunk

**178. Hová kerülhet a COMMIT az UNDO/REDO naplózás esetén? (2 pont)**

A <T,COMMIT> bejegyzés megelőzheti, de követheti is az adatbáziselemek lemezen történő bármilyen megváltozását.

**179. Adjunk meg egy példát UNDO/REDO naplózás esetén a lemezre írás sorrendjére! (6 pont)**

| Lépés | Tevékenység | t | M-A | M-B | D-A | D-B | Napló |
|---|---|---|---|---|---|---|---|
| 1) | | | | | | | <T,START> |
| 2) | READ(A,t) | 8 | 8 | | 8 | 8 | |
| 3) | t := t*2 | 16 | 8 | | 8 | 8 | |
| 4) | WRITE(A,t) | 16 | 16 | | 8 | 8 | <T,A,8,16> |
| 5) | READ(B,t) | 8 | 16 | 8 | 8 | 8 | |
| 6) | t := t*2 | 16 | 16 | 8 | 8 | 8 | |
| 7) | WRITE(B,t) | 16 | 16 | 16 | 8 | 8 | <T,B,8,16> |
| 8) | FLUSH LOG | | | | | | |
| 9) | OUTPUT(A) | 16 | 16 | 16 | 16 | 8 | |
| 10) | | | | | | | <T,COMMIT> |
| 11) | OUTPUT(B) | 16 | 16 | 16 | 16 | 16 | |

**180. Mi az UNDO/REDO naplózás esetén a helyreállítás 2 alapelve? (4 pont)**

REDO: A legkorábbitól kezdve állítsuk helyre minden befejezett tranzakció hatását.

UNDO: A legutolsótól kezdve tegyük semmissé minden be nem fejezett tranzakció tevékenységeit.

**181. Mi lehet probléma az UNDO/REDO naplózás esetén? (2 pont)**

Előfordulhat, hogy a tranzakció a felhasználó számára korrekten befejezettnek tűnik, de még a <T,COMMIT> naplóbejegyzés lemezre kerülése előtt fellépett hiba utáni helyreállítás során a rendszer a tranzakció hatásait semmissé teszi ahelyett, hogy helyreállította volna.

**182. Adjuk meg az UR2 szabályt az UNDO/REDO naplózás esetén? (2 pont)**

A <T,COMMIT> naplóbejegyzést azonnal lemezre kell írni, amint megjelenik a naplóban.

**183. Adjunk meg a működés közbeni ellenőrzőpont képzésének lépéseit UNDO/REDO naplózás esetén! (6 pont)**

1. **Írjunk a naplóba <START CKPT(T1,...,Tk)> naplóbejegyzést, ahol T1,...,Tk az aktív tranzakciók, majd írjuk a naplót lemezre.**
2. **Írjuk lemezre az összes piszkos puffert, tehát azokat, melyek egy vagy több módosított adatbáziselemet tartalmaznak. A helyrehozó naplózástól eltérően itt az összes piszkos puffert lemezre írjuk, nem csak a már befejezett tranzakciók által módosítottakat.**
3. **Írjunk <END CKPT> naplóbejegyzést a naplóba, majd írjuk a naplót lemezre.**

**184. Adjunk meg a működés közbeni mentés 5 lépését! (5 pont)**

1. **A <START DUMP> bejegyzés naplóba írása.**
2. **A REDO vag UNDO/REDO naplózási módnak megfelelő ellenőrzőpont kialakítása.**
3. **Az adatlemez(ek) teljes vagy növekményes mentése.**
4. **A napló mentése. A mentett naplórész tartalmazza legalább a 2. pontbeli ellenőrzőpont-képzés közben keletkezett naplóbejegyzéseket, melyeknek túl kell élniük az adatbázist hordozó eszköz meghibásodását.**
5. **<END DUMP> bejegyzés naplóba írása.**

**185. Az Oracle milyen naplózást valósít meg? (2 pont)**

Az Oracle az UNDO és a REDO naplózás egy speciális keverékét valósítja meg.

**186. Az Oracle mit használ UNDO naplózás céljára? (3 pont)**

A tranzakciók hatásainak semmissé tételéhez szükséges információkat a rollback szegmensek tartalmazzák. Minden adatbázisban van egy vagy több rollback szegmens, amely a tranzakciók által módosított adatok régi értékeit tárolja attól függetlenül, hogy ezek a módosítások lemezre íródtak vagy sem. A rollback szegmenseket használjuk az olvasási konzisztencia biztosítására, a tranzakciók visszagörgetésére és az adatbázis helyreállítására is.

**187. Az Oracle mit használ REDO naplózás céljára? (2 pont)**

A helyreállítás a napló (redo log) alapján történik A napló olyan állományok halmaza, amelyek az adatbázis változásait tartalmazzák, akár lemezre kerültek, akár nem. Két részből áll: az online és az archivált naplóból.

**188. Mit tartalmaz az Oracle rollback szegmense? (4 pont)**

A rollback szegmens rollback bejegyzésekből áll. Egy rollback bejegyzés többek között a megváltozott blokk azonosítóját (fájlsorszám és a fájlon belüli blokkazonosító) és a blokk régi értékét tárolja. A rollback bejegyzés mindig előbb kerül a rollback szegmensbe, mint ahogy az adatbázisban megtörténik a módosítás. Az ugyanazon tranzakcióhoz tartozó bejegyzések össze vannak láncolva, így könnyen visszakereshetők, ha az adott tranzakciót vissza kell görgetni.

**189. Milyen problémát kell megoldania a konkurencia-vezérlésnek? (4 pont)**

A tranzakciók közötti egymásra hatás az adatbázis inkonzisztensé válását okozhatja még akkor is, ha a tranzakciók külön-külön megőrzik a konzisztenciát, és rendszerhiba sem történt.

**190. Mit hívunk ütemezőnek? (2 pont)**

Az adatbázis-kezelő rendszer azon részét, ami a tranzakciós lépések szabályozásának feladatát végzi.

**191. Mit hívunk ütemezésnek? (2 pont)**

Az ütemezés egy vagy több tranzakció által végrehajtott lényeges műveletek időrendben vett sorozata, amelyben az egy tranzakcióhoz tartozó műveletek sorrendje megegyezik a tranzakcióban megadott sorrenddel.

**192. Milyen 2 módon biztosítja az ütemező a sorbarendezhetőséget? (2 pont)**

Várakoztat, megszakítást rendel el

**193. Mit hívunk konfliktuspárnak? (2 pont)**

Olyan egymást követő műveletpár az ütemezésben, amelynek ha a sorrendjét felcseréljük, akkor legalább az egyik tranzakció viselkedése megváltozhat.

**194. Milyen 3 esetben nem cserélhetjük fel a műveletek sorrendjét, mert inkonzisztenciát okozhatna? (3 pont)**

1, $r_i$ (X); $w_i$ (Y)

2, $w_i$ (X); $w_j$ (X)

3, $r_i$ (X); $w_j$ (X) és $w_i$ (X); $r_j$ (X)

**195. Mikor konfliktusekvivalens 2 ütemezés? (2 pont)**

Azt mondjuk, hogy két ütemezés konfliktusekvivalens, ha szomszédos műveleteinek nem konfliktusos cseréinek sorozatával az egyiket átalakíthatjuk a másikká.

**196. Mikor konfliktus-sorbarendezhető egy ütemezés? (2 pont)**

Azt mondjuk, hogy egy ütemezés konfliktus-sorbarendezhető, ha konfliktusekvivalens valamely soros ütemezéssel.

**197. Mi a konfliktus-sorbarendezhetőség elve? (3 pont)**

Nem konfliktusos cserékkel az ütemezést megpróbáljuk soros ütemezéssé átalakítani. Ha ezt meg tudjuk tenni, akkor az eredeti ütemezés sorbarendezhető volt, ugyanis az adatbázis állapotára való hatása változatlan marad minden nem konfliktusos cserével.

**198. Mi a kapcsolat a sorbarendezhetőség és a konfliktus-sorbarendezhetőség között? (2 pont)**

A konfliktus-sorbarendezhetőség elégséges feltétel a sorbarendezhetőségre.

**199. Az $r_1(A)$; $w_1(A)$; $r_2(A)$; $w_2(A)$; $r_1(B)$; $w_1(B)$; $r_2(B)$; $w_2(B)$; ütemezést alakítsuk soros ütemezéssé (5 pont)**

1. $r_1(A); w_1(A); r_2(A); w_2(A); r_1(B); w_1(B); r_2(B); w_2(B);$
2. $r_1(A); w_1(A); r_2(A); r_1(B); w_2(A); w_1(B); r_2(B); w_2(B);$
3. $r_1(A); w_1(A); r_1(B); r_2(A); w_2(A); w_1(B); r_2(B); w_2(B);$
4. $r_1(A); w_1(A); r_1(B); r_2(A); w_1(B); w_2(A); r_2(B); w_2(B);$
5. $r_1(A); w_1(A); r_1(B); w_1(B); r_2(A); w_2(A); r_2(B); w_2(B);$

**200. Adjunk példát sorbarendezhető, de nem konfliktus-sorbarendezhető ütemezésre (4 pont)**

Tekintsünk $T_1$, $T_2$ és $T_3$ tranzakciókat és egy soros ütemezésüket:

S: $w_1(Y)$; $w_2(Y)$; $w_2(X)$; $w_1(X)$; $w_3(X)$;

Mivel S soros ütemezés, ezért sorbarendezhető. Ugyanakkor mivel nem tudjuk felcserélni w1(X)-et w2(X)-szel, így cseréken keresztül nem lehet S2-t valamelyik soros ütemezéssé átalakítani. Tehát S sorbarendezhető, de nem konfliktus-sorbarendezhető.

**201. Mi a konfliktus-sorbarendezhetőség tesztelésének alapötlete? (2 pont)**

Ha valahol konfliktusban álló műveletek szerepelnek S-ben, akkor az ezeket a műveleteket végrehajtó tranzakcióknak ugyanabban a sorrendben kell előfordulniuk a konfliktusekvivalens soros ütemezésekben, mint ahogyan az S-ben voltak.

**202. Mikor mondjuk, hogy egy S ütemezés alapján $T_1$*megelőzi* $T_2$-t? (5 pont)**

Adott a $T_1$ és $T_2$, esetleg további tranzakcióknak egy S ütemezése. Azt mondjuk, hogy $T_1$ megelőzi $T_2$-t, ha van a $T_1$-ben olyan $A_1$ művelet és a $T_2$-ben olyan $A_2$ művelet, hogy:

1, $A_1$ megelőzi $A_2$-t S-ben

2, $A_1$ és $A_2$ ugyanarra az adatbáziselemre vonatkoznak

3, $A_1$ és $A_2$ közül legalább az egyik írás művelet

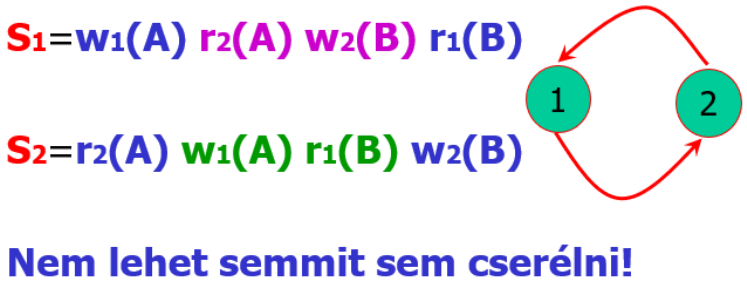
**203. Adjuk meg egy S ütemezéshez tartozó megelőzési gráf definícióját! (5 pont)**

A megelőzési gráf csúcsai az S ütemezés tranzakciói. Ha a tranzakciókat $T_i$-vel jelöljük, akkor a $T_i$-nek megfelelő csúcsot az i egész jelöli. Az i csúcsból a j csúcsba akkor vezet irányított él, ha: $T_i <_s T_j$.

**204. Milyen kapcsolat van a konfliktus-ekvivalencia és a megelőzési gráfok között? (4 pont)**

S1, S2 konfliktusekvivalens ➔gráf(S1) = gráf(S2), fordítva viszont nem igaz

**205. Adjunk példát arra, hogy két ütemezés megelőzési gráfja megegyezik, de nem konfliktus-ekvivalensek! (4 pont)**




**Nem lehet semmit sem cserélni!**

**206. Mit hívunk egy irányított, körmentes gráf esetében a csúcsok topologikus sorrendjének? (4 pont)**

Egy körmentes gráf csúcsainak topologikus sorrendje a csúcsok bármely olyan rendezése, amelyben minden a→b élre az a csúcs megelőzi a b csúcsot a topologikus rendezésben.

**207. Hogyan lehet tesztelni a megelőzési gráf alapján egy ütemezés konfliktus-sorbarendezhetőségét? (4 pont)**

Ha az S megelőzési gráf tartalmaz irányított kört, akkor S nem konfliktus-sorbarendezhető, ha nem tartalmaz irányított kört, akkor S konfliktus-sorbarendezhető, és a csúcsok bármelyik topologikus sorrendje megadja a konfliktusekvivalens soros sorrendet.

**208. Mi jellemző a passzív ütemezésre? (4 pont)**

- hagyjuk a rendszert működni

- az ütemezésnek megfelelő gráfot tároljuk

- egy idő után megnézzük, hogy van-e benne kör

- ha nincs, akkor szerencsénk volt, jó volt az ütemezés

**209. Mi jellemző az aktív ütemezésre és milyen 3 módszert lehet erre használni? (5 pont)**

Az ütemező beavatkozik és megakadályozza, hogy kör alakuljon ki.

Eszközei: zárak, időbélyegek, érvényesítés

**210. Mit hívunk a tranzakciók konzisztenciájának zárolási ütemező esetén? (2 pont)**

A tranzakció csak akkor olvashat vagy írhat egy elemet, ha már korábban zárolta azt, és még nem oldotta fel a zárat.

Ha egy tranzakció zárol egy elemet, akkor később azt fel kell szabadítania.

**211. Mit hívunk a zárolási ütemező jogszerűségének? (1 pont)**

Nem zárolhatja két tranzakció ugyanazt az elemet, csak úgy, ha az egyik előbb már feloldotta a zárat.

**212. Adjunk példát konzisztens tranzakciók jogszerű ütemezésére, ami mégsem sorbarendezhető! (6 pont)**

| $T_1$ | $T_2$ | A | B |
|---|---|---|---|
| $l_1(A)$; $r_1(A)$; | | 25 | |
| A := A+100; | | | |
| $w_1(A)$; $u_1(A)$; | | 125 | |
| | $l_2(A)$; $r_2(A)$; | 125 | |
| | A := A*2; | | |
| | $w_2(A)$; $u_2(A)$; | 250 | |
| | $l_2(B)$; $r_2(B)$; | | 25 |
| | B := B*2; | | |
| | $w_2(B)$; $u_2(B)$; | | 50 |
| $l_1(B)$; $r_1(B)$; | | | 50 |
| B := B+100; | | | |
| $w_1(B)$; $u_1(B)$; | | | 150 |

**213. Mit hívunk kétfázisú zárolásnak és szemléltessük rajzban is? (2 pont)**

Minden tranzakcióban minden zárolási művelet megelőzi az összes az összes zárfeloldási műveletet.


**214. Adjunk a tranzakciókra 2, az ütemezésre 1 feltételt, ami elegendő a konfliktus-sorbarendezhetőség bizonyítására! Milyen módon bizonyítható a tétel? (5 pont)**

Konzisztens, kétfázisú zárolású tranzakciók bármely S jogszerű ütemezését át lehet alakítani konfliktusekvivalens soros ütemezéssé. Bizonyítani S-ben részt vevő tranzakciók száma (n) szerinti indukcióval tudjuk.

**215. Mi a várakozási gráf és hogyan segít a holtpont felismerésében? (4 pont)**

A várakozási gráf csúcsai a tranzakciók, és akkor van él $T_i$-ből $T_j$-be, ha $T_i$ vár egy olyan zár elengedésére, amit $T_j$ tart éppen. Az ütemezés során egy adott pillanatban pontosan akkor nincs holtpont, ha az adott pillanathoz tartozó várakozási gráfban nincs irányított kör.

**216. Milyen két lehetőséggel védekezhetünk a holtpont ellen? (4 pont)**

1, Minden egyes tranzakció előre elkéri az összes zárat, ami neki kelleni fog. Ha nem kapja meg az összeset, akkor egyet sem kér el, el sem indul.

2, Feltesszük, hogy van egy sorrend az adategységeken és minden egyes tranzakció csak eszerint a sorrend szerint növekvően kérhet újabb zárakat. Itt lehet, hogy lesz várakozás, de holtpont biztos nem lesz.

**217. Mi a kiéheztetés problémája és milyen megoldás van rá? (2 pont)**

Többen várnak ugyanarra a zárra, de amikor felszabadul mindig elviszi valaki a tranzakció orra elől. Megoldás rá, ha adategységenként FIFO listában tartjuk a várakozókat, azaz mindig a legrégebben várakozónak adjuk oda a zárolási lehetőséget.

**218. Osztott és kizárólagos zárak esetén mit hívunk a tranzakció konzisztenciájának? (2 pont)**

Nem írhatunk kizárólagos zár fenntartása nélkül, és nem olvashatunk valamilyen zár fenntartása nélkül. Minden zárolást követnie kell egy ugyanannak az elemnek a zárolását feloldó műveletnek.

**219. Osztott és kizárólagos zárak esetén mit hívunk az ütemezés jogszerűségének? (2 pont)**

Egy elemet vagy egyetlen tranzakció zárol kizárólagosan, vagy több is zárolhatja osztottan, de a kettő egyszerre nem lehet.

**220. Osztott és kizárólagos zárak esetén adjunk meg feltételeket az ütemezés konfliktus-sorbarendezhetőségére? (4 pont)**

Konzisztens 2PL tranzakciók jogszerű ütemezése konfliktus-sorbarendezhető.

**221. Osztott és kizárólagos zárak esetén adjuk meg a kompatibilitási mátrixot!(4 pont)**

| | | S | X |
|---|---|---|---|
| Ha ilyen zár van már kiadva | S | igen | nem |
| | X | nem | nem |

Megkaphatjuk-e ezt a típusú zárat?

**222. Többmódú zárak kompatibilitási mátrixa segítségével hogyan definiáljuk a megelőzési gráfot? (5 pont)**

Az ütemező a kompatibilitási mátrix alapján dönti el, hogy egy ütemezés/zárkérés legális-e, illetve ez alapján várakoztatja a tranzakciókat. Minél több az Igen a mátrixban, annál kevesebb lesz a várakoztatás. A mátrix alapján keletkező várakozásokhoz elkészített várakozási gráf segítségével az ütemező kezeli a holtpontot.

**223. Többmódú zárak esetén a megelőzési gráf segítségével hogy lehet eldönteni a sorbarendezhetőséget? (3 pont)**

Egy csak zárkéréseket és zárelengedéseket tartalmazó jogszerű ütemezés sorbarendezhető akkor és csak akkor, ha a kompatibilitási mátrix alapján felrajzolt megelőzési gráf nem tartalmaz irányított kört.

**224. Adjunk példát arra, hogy egy zárolási ütemező elutasít sorbarendezhető ütemezést? (4 pont)**

Tekintsük az $l_1(A)$; $r_1(A)$; $u_1(A)$; $l_2(A)$; $r_2(A)$; $u_2(A)$; $l_1(A)$; $w_1(A)$; $u_1(A)$; $l_2(B)$; $r_2(B)$; $u_2(B)$ ütemezést.

Ha megnézzük az írás/olvasás műveleteket ($r_1(A)$; $r_2(A)$; $w_1(A)$; $r_2(B)$) akkor látszik, hogy az ütemezés hatása azonos a $T_2T_1$ soros ütemezés hatásával, vagyis ez egy sorbarendezhető ütemezés zárak nélkül.


Ha viszont felrajzoljuk a zárakra vonatkozó megelőzési gráfot, akkor irányított kört fog tartalmazni, és ezért elvetnénk.

**225. Adjunk feltételt az ütemezés sorbarendezhetőségére tetszőleges zármodellben! (4 pont)**

Ha valamilyen zármodellben egy jogszerű ütemezésben minden tranzakció követi a 2PL-t, akkor az ütemezéshez tartozó megelőzési gráf nem tartalmaz irányított kört, azaz az ütemezés sorbarendezhető.

**226. Mikor mondjuk, hogy egyik zár erősebb a másiknál? (4 pont)**

Azt mondjuk, hogy L2 erősebb L1-nél, ha a kompatibilitási mátrixban L2 sorában/oszlopában minden olyan pozíción „NEM” áll, amelynél L1 sorában/oszlopában „NEM” áll.

**227. Adjuk meg a módosítási zár kompatibilitási mátrixát és értelmezzük röviden!(4 pont)**

| | S | X | U |
|---|---|---|---|
| S | igen | nem | igen |
| X | nem | nem | nem |
| U | nem | nem | nem |

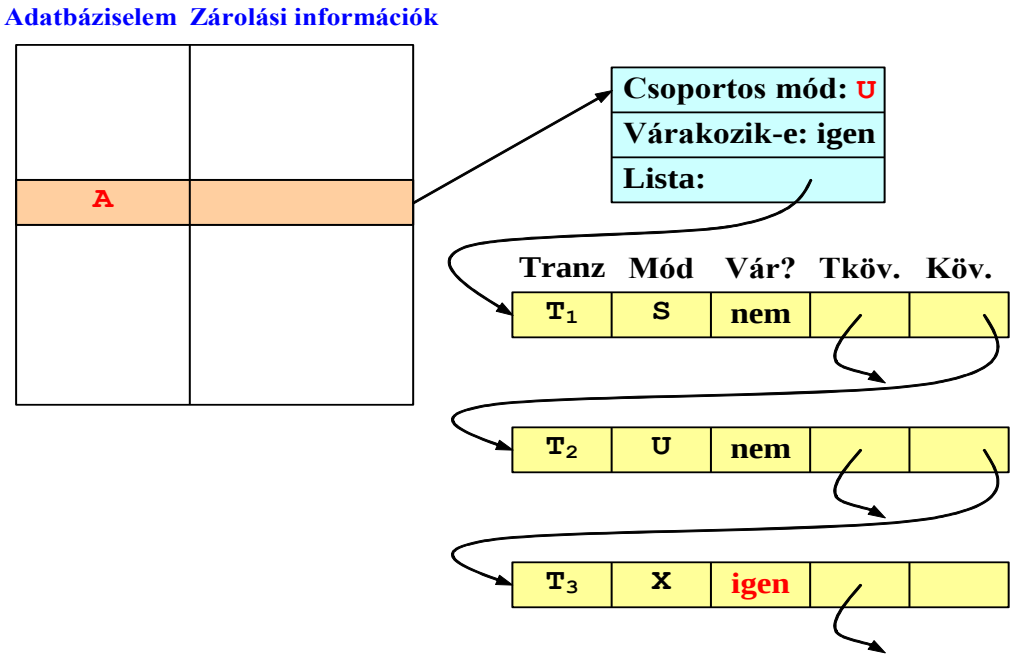
Az U módosítási zár úgy néz ki, mintha osztott zár lenne amikor kérjük, és úgy néz ki, mintha kizárólagos zár lenne, amikor már megvan.

**228. Mi az $inc_i$(X) művelet és adjuk meg a növelési zár kompatibilitási mátrixát! (4 pont)**

A $T_i$ tranzakció megnöveli az x adatbáziselemet valamely konstanssal (a konstans pontos értékének nincs jelentősége).

| | S | X | I |
|---|---|---|---|
| S | igen | nem | nem |
| X | nem | nem | nem |
| I | nem | nem | igen |

**229. Adjunk meg a zártábla egy lehetséges formáját, a mezők tartalmát magyarázzuk is el! (8 pont)**




- Az összes olyan tranzakciót leíró lista, amelyek vagy jelenleg zárolják A-t, vagy A zárolására várakoznak.
- A csoportos mód az adatelemre kiadott legerősebb zár.
- A várakozási bit azt adja meg, hogy van-e legalább egy tranzakció, amely az A zárolására várakozik.

**230. A zárfeloldások sorrendje milyen elvek alapján történhet? (3 pont)**

- Első beérkezett kiszolgálása először

- Elsőbbségadás az osztott záraknak

- Elsőbbségadás a felminősítésnek

**231. Hierarchikus adatok esetén mi a figyelmeztető zárak használatának három alapelve? (3 pont)**

- A két zárnak megfelelő figyelmeztető zárakat kérünk az útvonal mentén a gyökérből kiindulva az adatelemig.

- Addig nem megyünk lejjebb, amíg a figyelmeztető zárat meg nem kapjuk.

- Így a konfliktusos helyzetek alsóbb szintekre kerülnek a fában.

**232. Hierarchikus adatok esetén adjuk meg az osztott, kizárólagos és figyelmeztető zárakra vonatkozó kompatibilitási mátrixot? (4 pont)**

**SOR: Ha ilyen zár van már kiadva**

| | IS | IX | S | X |
|---|---|---|---|---|
| IS | igen | igen | igen | nem |
| IX | igen | igen | nem | nem |
| S | igen | nem | igen | nem |
| X | nem | nem | nem | nem |

**Oszlop: Megkaphatjuk-e ezt a típusú zárat?**

**233. Hierarchikus adatok esetén miért vezetjük be az SIX zártípust és mi jellemző rá? (4 pont)**

Azért vezetjük be az SIX zártípust, mivel IS < IX és S < X, de IX és S nem összehasonlítható. Azt jelenti, hogy ugyanaz a tranzakció S és IX zárat is tett egy adatelemre. Ekkor SIX mindkettőnél erősebb, de ez a legkisebb ilyen.

**234. Adjuk meg a csoportos móddal kiegészített figyelmeztető zárakra vonatkozó kompatibilitási mátrixot! (5 pont)**

Kérés

| Zárolás | IS | IX | S | SIX | X |
|---|---|---|---|---|---|
| IS | T | T | T | T | F |
| IX | T | T | F | F | F |
| S | T | F | T | F | F |
| SIX | T | F | F | F | F |
| X | F | F | F | F | F |

**T="igen"**

**F="nem"**

**235. Mit hívunk nem ismételhető olvasásnak és mi a probléma vele? (4 pont)**

Tegyük fel, hogy van egy $T_1$ tranzakció, amely egy adott feltételnek eleget tevő sorokat válogat ki egy relációból. Ezután hosszas számításba kezd, majd később újra végrehajtja a fenti lekérdezést. Tegyük fel továbbá, hogy a lekérdezés két végrehajtása között egy $T_2$ tranzakció módosít vagy töröl a táblából néhány olyan sort, amely eleget tesz a lekérdezés feltételének. A $T_1$ tranzakció lekérdezését ilyenkor nem ismételhető (fuzzy) olvasásnak nevezzük. A nem ismételhető olvasással az a probléma, hogy mást eredményez a lekérdezés másodszori végrehajtása, mint az első.

**236. Mit hívunk fantom soroknak? (3 pont)**

Tegyük fel, hogy T tranzakció beszúr olyan sorokat, amelyek eleget tesznek egy lekérdezés feltételének. A lekérdezés másodszori futtatásakor más eredményt kapunk, mint első alkalommal. Ennek az az oka, hogy most olyan sorokat is figyelembe kellett venni, melyek az első futtatáskor még nem is léteztek. Az ilyen sorokat nevezzük fantomoknak.

**237. Mikor követi egy tranzakció a faprotokollt? Adjuk meg a faprotokoll 4 szabályát! (4 pont)**

1, Az első zárat a tranzakció bárhova elhelyezheti.

2, A későbbiekben azonban csak akkor kaphat zárat A adategységen, ha ekkor zárja van A apján.

3, Zárat bármikor fel lehet oldani.

4, Nem lehet újra zárolni. Tehát, ha $T_i$ elengedte egy A adategység zárját, akkor később nem kérhet rá újra.

**238. Hierarchiák, például B*-fa elemeinek zárolása esetén milyen feltétel adható az ütemezés sorbarendezhetőségére? (4 pont)**

Ha minden tranzakció követi a faprotokollt egy jogszerű ütemezésbe, akkor az ütemezés sorbarendezhető lesz, de nem feltétlenül lesz 2PL.

**239. Mi az időbélyegzési módszer lényege? Használunk-e ilyenkor zárakat? (4 pont)**

Minden tranzakcióhoz hozzárendelünk egy időbélyegzőt. Minden adatbáziselem utolsó olvasását és írását végző tranzakció időbélyegzőjét rögzítjük, és összehasonlítjuk ezeket az értékeket, hogy biztosítsuk, hogy a tranzakciók időbélyegzőinek megfelelő soros ütemezés ekvivalens legyen a tranzakciók aktuális ütemezésével. Nem használunk zárakat.

**240. Adjunk meg három jellemzőt az Oracle konkurenciavezérlésére vonatkozóan! (3 pont)**

Az Oracle alkalmazza a kétfázisú zárolást, a figyelmeztető protokollt és a többváltozatú időbélyegzőket.

**241. Milyen olvasási konzisztenciát biztosít az Oracle és mivel éri ezt el? (3 pont)**

Az Oracle minden lekérdezés számára biztosítja az utasítás szintű olvasási konzisztenciát, továbbá kérhetjük a tranzakció összes lekérdezése számára a tranzakció szintű olvasási konzisztenciát. Ennek biztosítása céljából az Oracle a rollback szegmensben található információkat használja fel.

**242. Adjuk meg az SQL92 ANSI/ISO szabványban szereplő tranzakciós elkülönítési szinteket! (4 pont)**

nem olvasásbiztos, olvasásbiztos, megismételhető olvasás, sorbarendezhető

**243. Mi jellemző a nem olvasásbiztos elkülönítési szintre a piszkos, fantom, nem ismételhető olvasásokra vonatkozóan? (3 pont)**

piszkos olvasás – lehetséges, fantomok olvasása – lehetséges, nem ismételhető olvasás - lehetséges

**244. Mi jellemző az olvasásbiztos elkülönítési szintre a piszkos, fantom, nem ismételhető olvasásokra vonatkozóan? (3 pont)**

piszkos olvasás – nem lehetséges, fantomok olvasása – lehetséges, nem ismételhető olvasás - lehetséges

**245. Mi jellemző a megismételhető olvasás elkülönítési szintre a piszkos, fantom, nem ismételhető olvasásokra vonatkozóan? (3 pont)**

piszkos olvasás – nem lehetséges, fantomok olvasása – lehetséges, nem ismételhető olvasás – nem lehetséges

**246. Mi jellemző a sorbarendezhető elkülönítési szintre a piszkos, fantom, nem ismételhető olvasásokra vonatkozóan? (3 pont)**

piszkos olvasás – nem lehetséges, fantomok olvasása – nem lehetséges, nem ismételhető olvasás – nem lehetséges

**247. Milyen DML szintű zárakat használ az Oracle? (2 pont)**

DML-zárakat két szinten kaphatnak a tranzakciók:

- sorok szintjén

- teljes táblák szintjén

**248. Milyen zártípusokat használ az Oracle sorokra és táblákra? (6 pont)**

Sorok szintjén csak egyféle zármód létezik: kizárólagos (írási – X)

Táblák szintjén ötféle zármódot különböztetünk meg:

1, row share (RS) vagy subshare (SS)

2, row exclusive (RX) vagy subexclusive (SX)

3, share (S)

4, share row exclusive (SRX) vagy share-subexclusive (SSX)

5, exclusive (X)

*2023/24 őszi félév ellenőrző kérdései alapján (Kidolgozta: Krascsenits Erik)*

https://people.inf.elte.hu/kiss/15ab2/13ab2osz.htm