# Adatbázisok 2 - Ellenőrző kérdések

**1. Mit hívunk statikus, és mit dinamikus adatbázisnak? (1 pont)**
Egy statikus adatbázis esetében ritkábban fordulnak elő módosítások, a lekérdezések gyorsasága fontosabb. Dinamikus adatbázis esetén gyakran hajtunk végre módosítást, lekérdezést ritkán végzünk.

**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 és kiírás költsége arányos a háttértároló és a memória között mozgatott blokkok számával.

**4. Adjuk meg az alábbi paraméterek jelentését! l, b, B, T, bf, M, I(A) (7 pont)**
A paramétereket az indexelés költségének méréséhez vezettük be: l – rekordméret
b – blokkméret (bájtokban)
B – a fájl mérete blokkokban ( B = ⌈T/bf⌉ )
T – (tuple) rekordok száma
bf – blokkolási faktor (mennyi rekord fér el egy blokkban: bf = ⌊b/l⌋ (alsó egészrész)) M – memória mérete blokkokban
I(A) – képméret, az A oszlopban szereplő különböző értékek száma: $\mathrm{I(A)} = |\Pi_A(\mathrm{R})|$

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

$$B(R \times S) = \big(T(R) * T(S)\big) * \frac{I(R) + I(S)}{b} =$$
$$\left(T(S) * T(R) * \frac{I(R)}{b}\right) + \left(T(R) * T(S) * \frac{I(S)}{B}\right) =$$
$$T(S) * B(R) + T(R) * B(S)$$

**6. Mit jelent az egyenletességi feltétel? (1 pont)**
Fel szoktuk tenni, hogy az A=A feltételnek eleget tevő rekordokból nagyjából egyforma számú rekord szerepel. Ezt nevezzük egyenletességi feltételnek.

**7. Mekkora adategységet olvas az író-olvasó fej? (1 pont)**
Az író-olvasó fej nagyobb adategységeket (blokkokat) olvas be.

**8. Mitől függhet a blokkméret? (1 pont)**
A blokkméret függhet az operációs rendszertől, hardvertől, adatbázis-kezelőtől.

**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 (heap)
- hasító index (hash)
- 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)**
Kupac szervezés esetén legrosszabb esetben a keresési költsége B (tárméret).

**12. Kupac szervezés esetén mennyi a beszúrás költsége? (1 pont)**
- utolsó blokkba tesszük a rekordot, 1 olvasás + 1 írás
- módosítás: 1 keresés + 1 írás
- törlés: 1 keresés + 1 írás

**13. Mit mond meg a h(x) hasító függvény értéke? (1 pont)**
A h(x) hasító függvény értéke megmondja, 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)**
Egy hasító függvény akkor jó, ha nagyjából egyforma hosszú blokkláncok keletkeznek, azaz egyenletesen sorolja be a rekordokat. Jó hasító függvény esetén a blokklánc B/K blokkból áll.

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

**16. Ha túl nagynak választjuk a K-t hasításkor, akkor ez milyen problémát okozhat? (1 pont)**
Nagy K esetén sok olyan blokklánc lehet, amely egy blokkból fog állni és a blokkban is csak 1 rekord lesz. Ekkor a keresési idő: 1 blokkbeolvasás, de B helyett T számú blokkban tároljuk az adatokat.

**17. Milyen keresésre nem jó a hasító indexelés? (1 pont)**
Intervallumos (a < A < b) típusú keresésre nem jó.

**18. Mit jelent a dinamikus hasító indexelés és milyen két fajtáját ismerjük? (3 pont)**
Olyan hasító indexelés, ahol előre nem rögzítjük a kosarak számát, a kosarak száma beszúráskor, törléskor változhat. Két fajtája van: kiterjeszthető és lineáris.

**19. Kiterjeszthető hasítás esetén a h(K) érték alapján melyik kosárba kerül a rekord? (2 pont)**
A h(K) k hosszú kódnak vegyük az i hosszú elejét, és azt kosarat, amelynek kódja a h(K) kezdő szelete. Ha van hely a kosárban, tegyük bele a rekordot, ha nincs, akkor nyissunk egy új kosarat, és a következő bit alapján osszuk ketté a telített kosár rekordjait. Ha ez a bit mindegyikre megegyezik, akkor a következő bitet vesszük a szétosztáshoz, és így tovább.

**20. Milyen probléma keletkezhet kiterjeszthető hasító index esetén és mi rá a megoldás? (2 pont)**
Ha az új sorok hasító értékének eleje sok bitben megegyezik, akkor hosszú ágak keletkeznek, nem lesz kiegyensúlyozva a fa. Tegyük teljessé a bináris gráfot. A gráfot egy tömbbel ábrázoljuk. Ekkor minden kosár azonos szinten 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 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, feltéve, 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)**
- beolvassuk a középső blokkot
- ha nincs benne az A=a értékű rekord, eldöntjük, hogy a blokklánc második felében, vagy az első felében szerepelhet-e egyáltalán
- beolvassuk a felezett blokklánc középső blokkját
- 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)**
$\log_2(\mathrm{B})$

**25. Mennyi a keresési költség rendezett mező esetében, ha gyűjtő blokkokat is használunk? (1 pont)**
Ha G a gyűjtő mérete, akkor az összköltség: $\log_2(\mathrm{B}-\mathrm{G})+\mathrm{G}$.

**26. Mennyi a keresési költség rendezett mező esetében, ha minden blokkot félig üresen hagyunk? (1 pont)**
$\log_2(2*\mathrm{B}) = 1+\log_2(\mathrm{B})$

**27. Milyen mindig az indexrekord szerkezete? (1 pont)**
Az indexrekord szerkezete (a,p), ahol a egy érték az indexelt oszlopban, p egy blokkmutató, arra a blokkra mutat, amelyben az A=a értékű rekordot találjuk.

**28. Adjuk meg az elsődleges index 5 jellemzőjét! (5 pont)**
- főfájl is rendezett
- csak 1 elsődleges indexet lehet megadni
- elég a főfájl minden blokkjának legkisebb rekordjához készíteni indexrekordot
- indexrekordok száma: T(I) = B (ritka index)
- indexrekordokból sokkal több fér egy blokkba, mint a főfájl rekordjaiból. bf(I) >> bf, azaz az indexfájl sokkal kisebb rendezett fájl mint a főfájl: B(I) = B / bf(I) << B = T / bf.

**29. Mit hívunk fedőértéknek? (1 pont)**
Az indexfájlban nem szerepel minden érték, ezért csak fedő értéket kereshetünk, a legnagyobb olyan indexértéket, amely a keresett értéknél kisebb vagy egyenlő

**30. Mennyi a keresési költség elsődleges index esetén? (1 pont)**
$1 + \log_2(\mathrm{B(I)})$

**31. Adjuk meg a másodlagos index 5 jellemzőjét! (5 pont)**
- főfájl rendezetlen (az indexfájl mindig rendezett)
- több másodlagos indexet is meg lehet adni
- a főfájl minden rekordjához kell készíteni indexrekordot
- indexrekordok száma: T(I) = T (sűrű index)
- indexrekordból sokkal több fér egy blokkba, mint a főfájl rekordjaiból: bf(I) >> bf, azaz az indexfájl sokkal kisebb rendezett fájl mint a főfájl: B(I) = T / bf(I) << B = T / bf.

**32. Hogyan keresünk a másodlagos indexben és mennyi a keresés költsége? (5 pont)**
- az indexben keresés az index rendezettsége miatt bináris kereséssel történik: $\log_2(\mathrm{B(I)})$
- a talált indexrekordban szereplő blokkmutatónak megfelelő blokkot még be kell olvasni
- $1 + \log_2(\mathrm{B(I)}) << \log_2(\mathrm{B})$ (rendezett eset)
- az elsődleges indexnél rosszabb a keresési idő, mert több az indexrekord.

**33. Mit hívunk klaszterszervezésű táblának? (1 pont)**
Egy tábla esetén egy A oszlopra az azonos A-értékű sorok fizikailag egymás után blokkokban helyezkednek el.

**34. Mit hívunk klaszterindexnek? (1 pont)**
Klaszterszervezésű fájl esetén index az A oszlopra.

**35. Mikor mondjuk, hogy 2 tábla klaszterszervezésű? (1 pont)**
Ha a közös oszlopokon egyező sorok egy blokkban, vagy fizikailag egymás utáni blokkban helyezkednek el.

**36. Ha t szintű indexet használunk, mennyi a keresési költség blokkműveletek számában mérve? (1 pont)**
$\log_2(\mathrm{B(I^{(t)})}) + \mathrm{t}$

**37. Ha t szintű indexet használunk, a legfelső szinten milyen keresést használunk? (1 pont)**
A t-ik szinten $(\mathrm{I^{(T)}})$ bináris kereséssel keressük meg a fedő indexrekordot

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

**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) |

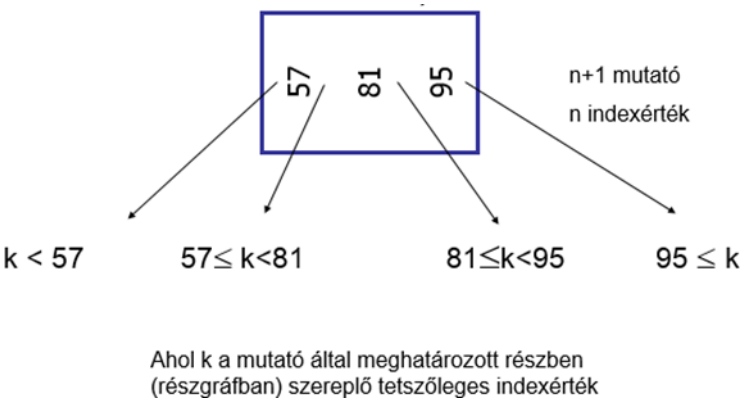
**41. Ha t szintű indexet használunk, és a legfelső szint 1 blokkból áll, abból milyen egyenlet következik és mi a megoldása t-re? (2 pont)**

t-ik szinten 1 blokk: $1 = B/bf(I)^t$, azaz $t = \log_{bf(I)}B < \log_2(B)$ azaz jobb a rendezett fájlszervezésnél

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

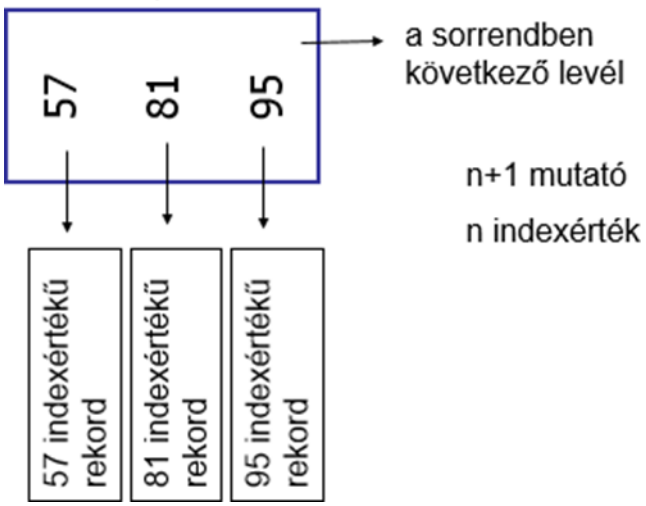
Minden blokk legalább 50%-ban telített. A szerkezeten kívül a telítettséget biztosító karbantartó algoritmusokat is beleértjük.

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

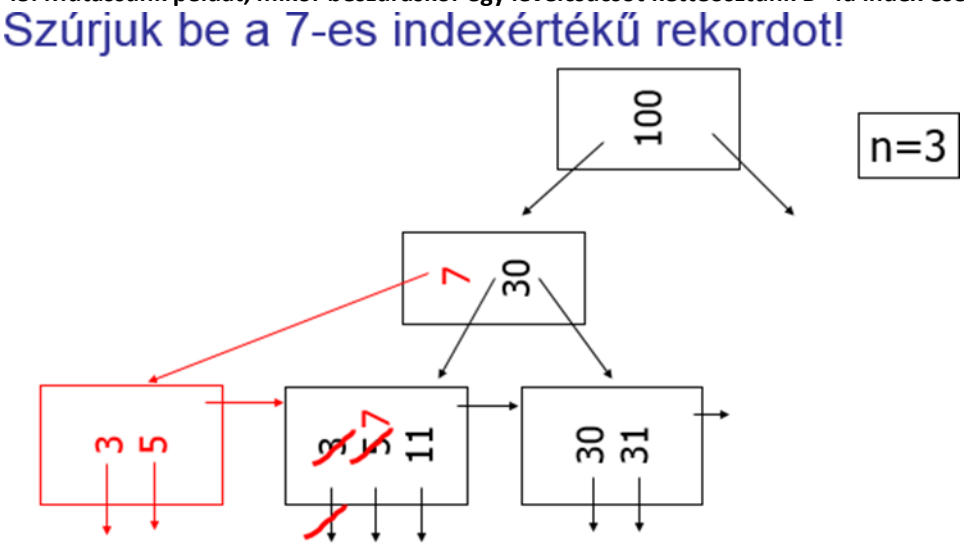



Ahol k a mutató által meghatározott részben (részgráfban) szereplő tetszőleges indexérték

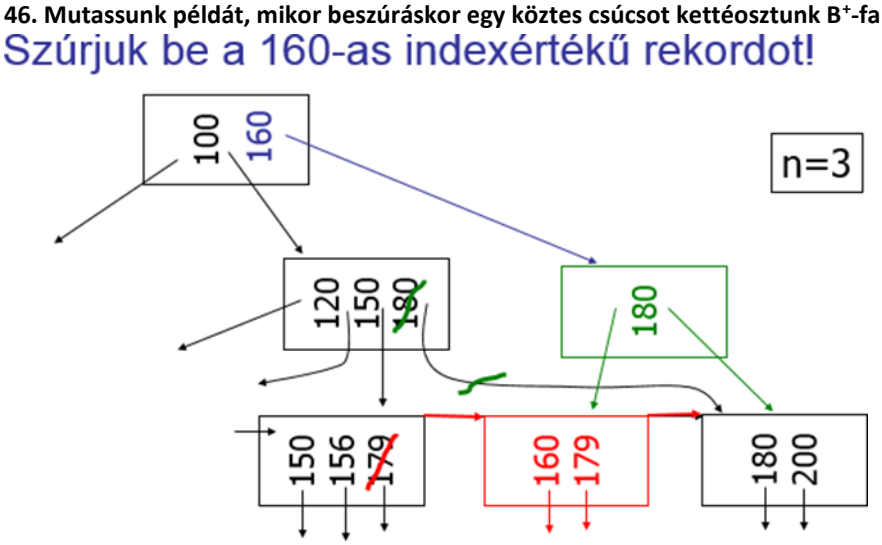
**44. Egy példa alapján szemléltessük a levél csúcs jellemzőit B⁺-fa index esetén! (5 pont)**



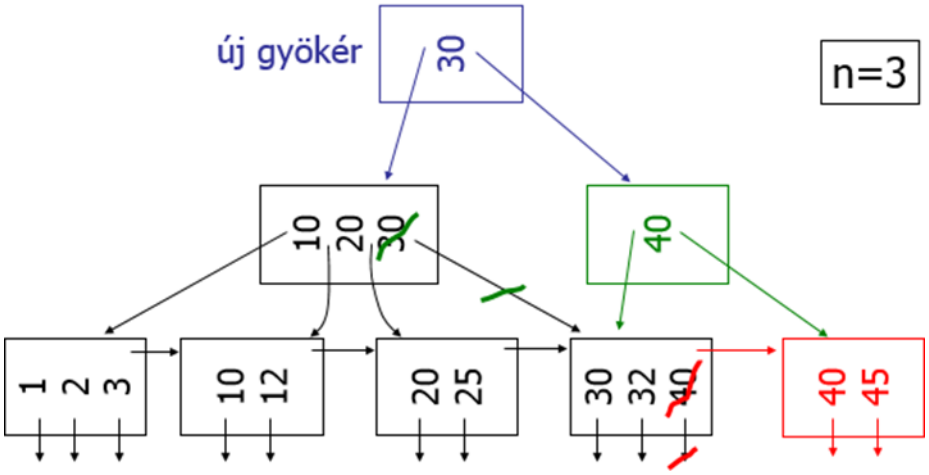
**45. Mutassunk példát, mikor beszúráskor egy levélcsúcsot kettéosztunk B$^+$-fa index esetén! (5 pont)**




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

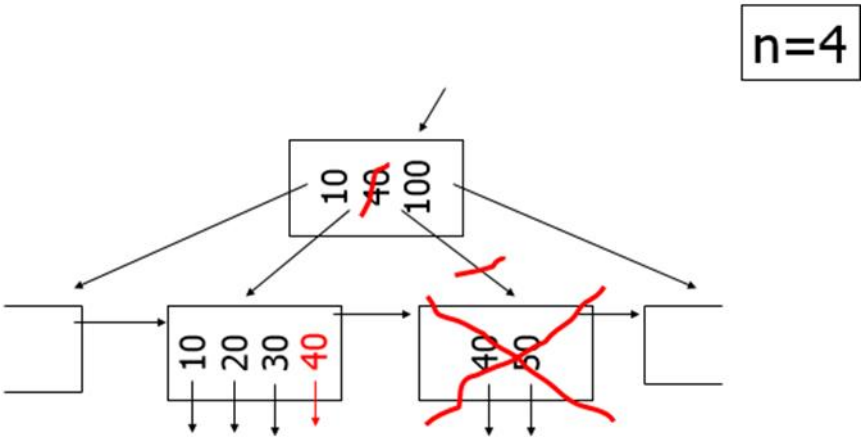



**47. Mutassunk példát, mikor beszúráskor nő a B$^+$-fa index magassága! (5 pont)**



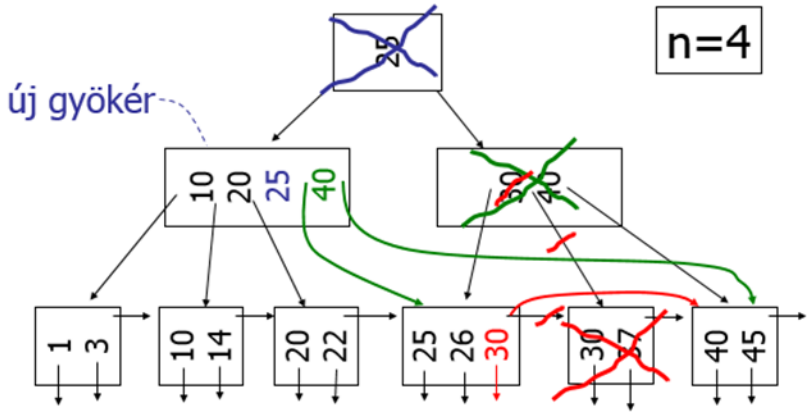
**48. Mutassunk példát, mikor törléskor megszüntetünk egy levélcsúcsot $B^+$-fa index esetén! (5 pont)**

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!




**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');
```

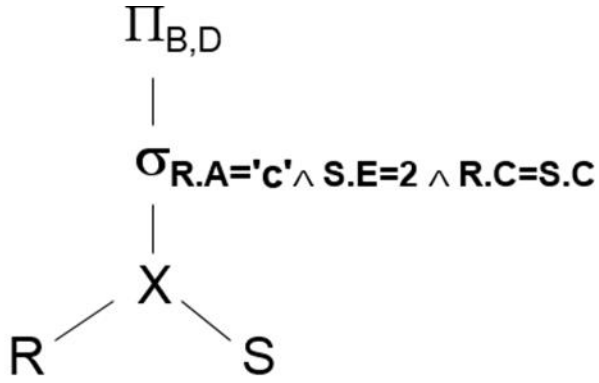
| status = 'married' | | region = 'central' | | region = 'west' | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | 0 | | 0 | | 0 | | 0 | | 0 |
| 1 | | 1 | | 0 | | 1 | | 1 | | 1 |
| 1 | AND | 0 | OR | 1 | = | 1 | AND | 1 | = | 1 |
| 0 | | 0 | | 1 | | 0 | | 1 | | 0 |
| 0 | | 1 | | 0 | | 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 táblákra vonatkozó paraméterek, statisztikák, indexek 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)**
Az eljárás heurisztikus, tehát 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)**
Unáris csúcsok, melyeknek csak egy gyerekük van: ($\sigma$, $\Pi$, $\rho$)

**60. A relációs algebrai kifejezésfában melyek a bináris csúcsok? (3 pont)**
Bináris csúcsok, melyeknek két gyereke van: (-, $\cup$, $\times$)

**61. A relációs algebrai kifejezésfában mik a levélcsúcsok? (2 pont)**
A levélcsúcsok konstans relációk, vagy relációs változók lehetnek.

**62. Mit hívunk ekvivalens relációs algebrai kifejezéseknek? (3 pont)**
E1(r1,…,rk) és E2(r1,…,rk) relációs algebrai kifejezések ekvivalensek ($E1 \cong E2$), 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 adhatunk 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)**
$\boldsymbol{E1 \times E2 \cong E2 \times E1}$
$\boldsymbol{E1 \,|\times|\, E2 \cong E2 \,|\times|\, E1}$
$\boldsymbol{E1 \,|\times|\, E2 \cong E2 \,|\times|\, E1}$
$\Theta \qquad \Theta$

**65. Adjuk meg a relációs algebrai optimalizálás asszociatív szabályait! (3 pont)**
$\boldsymbol{(E1 \times E2) \times E3 \cong E1 \times (E2 \times E3)}$
$\boldsymbol{(E1 \,|\times|\, E2) |\times|\, E3 \cong E1 |\times|\, (E2 \,|\times|\, E3)}$
$\boldsymbol{(E1 \,|\times|\, E2) |\times|\, E3 \cong E1 |\times|\, (E2 \,|\times|\, E3)}$
$\Theta \quad \Theta \qquad \Theta \quad \Theta$

**66. Adjuk meg a vetítésre vonatkozó összevonási, bővítés szabályt relációs algebrai optimalizálás esetén! (2 pont)**
Legyen $\underline{A}$ és $\underline{B}$ két részhalmaza a 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 $\boldsymbol{\Pi_{\underline{A}}(\sigma_F(E)) \cong \sigma_F(\Pi_{\underline{A}}(E))}$.

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

- Általánosabban: 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)**

- Speciálisan: 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 rel. algebrai optimalizálás esetén! (3 pont)**

- Általánosabban: Legyen F1 az E1 reláció oszlopainak egy részhalmazán értelmezett kiválasztási feltétel, legyen F2 az E1×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 $\sigma_F(E1 |\times| E2) \cong \sigma_F(E1) |\times| \sigma_F(E2)$.

**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)**

Megjegyzés: A vetítés és kivonás nem cserélhető fel, azaz $\Pi_A(E1 - E2) \neq \Pi_A(E1) - \Pi_A(E2)$. Például:

E1:

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

E2:

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

esetén $\Pi_A(E1 - E2)$:

| A |
|---|
| 0 |

míg

$\Pi_A(E1) - \Pi_A(E2) = \varnothing$

114

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

1. Minél hamarabb szelektáljunk
2. Próbáljunk természetes összekapcsolásokat képezni
3. Vonjuk össze az egymás utáni unáris műveleteket
4. Keressünk közös részkifejezéseket

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

Mert így a részkifejezések várhatóan kisebb relációk lesznek.

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

Az összekapcsolás hatékonyabban kiszámolható, mint a szorzatból történő kiválasztás.

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

Mert így csökken 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)**

Mert így csökken a műveletek száma, és általában a kiválasztás kisebb relációt eredményez, mint a vetítés.

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

Az input a relációs algebrai kifejezés kifejezésfája, az output pedig az optimalizált kifejezésfa optimalizált kiértékelése.

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

Bontsuk fel a kiválasztásokat: $\sigma F1 \wedge \ldots \wedge Fn(E) \cong \sigma F1(\ldots(\sigma Fn(E)))$

**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. Hagyjuk el a triviális vetítéseket, azaz az olyanokat, amelyek az argumentum reláció összes attribútumára vetítenek.

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

A relációs változóra, vagy konstans relációra közvetlenül egymás után alkalmazott kiválasztásokat, vagy vetítéseket vonjuk össze egy kiválasztássá, vagy egy vetítéssé, vagy egy kiválasztás utáni vetítéssé, ha lehet. Ezzel megkaptuk az optimalizált kifejezésfát.

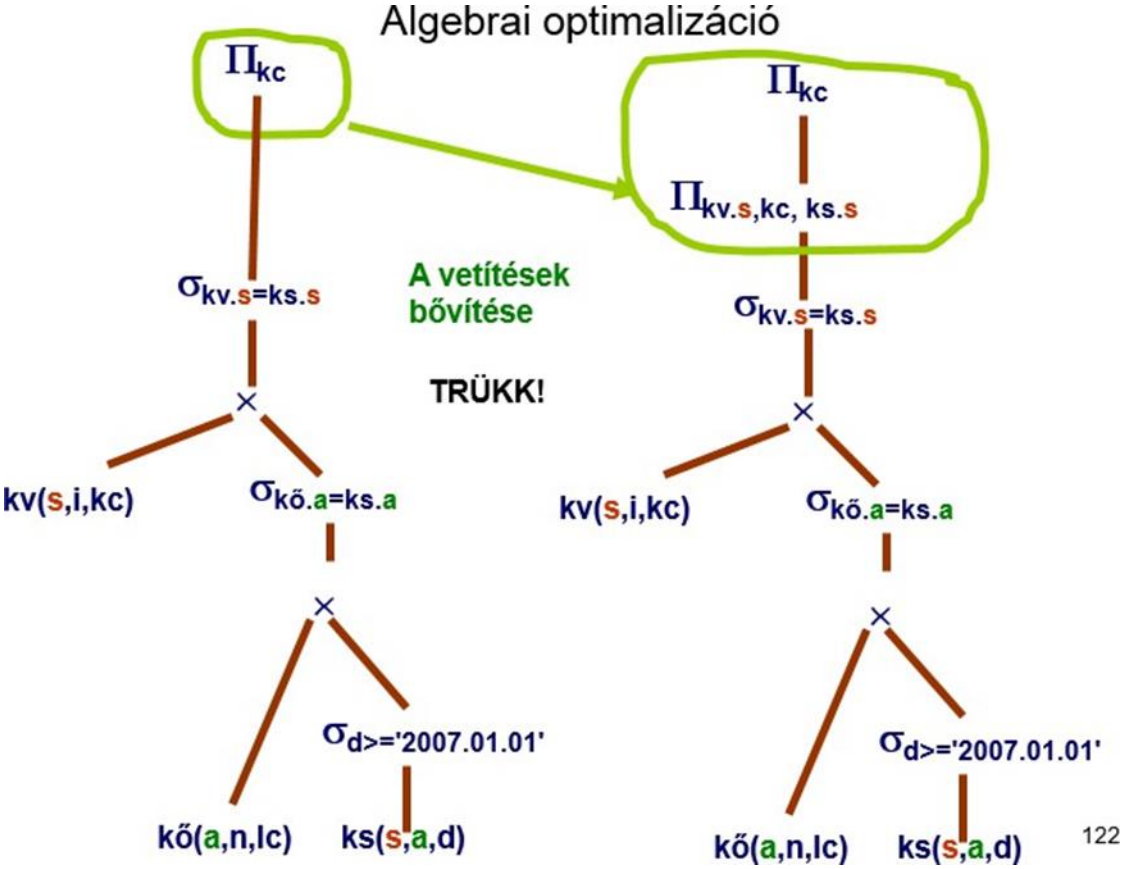
**88. Mi a relációs algebrai optimalizálás algoritmusának 5. lépése (az alkalmazott szabályok felsorolása nélkül)? (4 pont)**

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 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 optimális kiértékeléshez az 5. lépésben kapott fát értékeljük ki alulról felfelé haladva, tetszőleges sorrendben.

**90. Adjunk meg egy példát, amiben a vetítések bővítése trükköt alkalmazzuk! (8 pont)**




**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) = $N_R/V(A, R)$, ahol $N_R$ az R-ben szereplő rekordok száma, V(A,R) pedig az A attribútum különböző 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)**

Átlagos költség: $|log_2 B_r| + m$

Ahol: $m = \lceil SC(A, R)/F_R \rceil - 1$

**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)**

Egy rekord esetén: $HT_i + 1$

Több rekord esetén: $HT_i + \lceil SC(A, R)/F_R \rceil$

**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)**

Legrosszabb esetben az átlagos keresési költség: $HT_I + SC(A, R)$

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

– végezzünk egyszerű kiválasztást a legkisebb költségű θi-re
  • a fennmaradó θ feltételek szerint szűrjük az eredményt
– több index
  • válasszuk ki a θi-khez tartozó indexeket
  • keressünk az indexekben és adjuk vissza a RID-ket
  • válasz: 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 cid FROM takes
- π-hez szükséges a duplikált értékek kiszűrése
- rendezés
halmazműveletekhez ki kell szűrni a duplikált értékeket
- R ∩ S
- R ∪ 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)**

Sok művelet hatékony kiértékelésére.

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

Rendezett futamok összefésülésére.

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

Ha N > M. Jelölések?

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

A költség $2 * B_R$ , ahol $B_R$ az R lapjainak száma.

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

Kezdetben $|\frac{B_R}{M}|$ összevonandó futam van, ahol $B_R$ az R lapjainak száma, M pedig?

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

Az összes menet száma $\left\lceil log_{M-1}(\frac{B_R}{M}) \right\rceil$. Jelölések?

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

Minden menetben $2 * B_R$ lapot olvasunk, ahol $B_R$ az R lapjainak száma.

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

Teljes költség: $2 * B_R + 2 * B_R * |log_{M-1}(\frac{B_R}{M})| - B_R$. Jelölések?

**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) összekapcsolás
- Blokk-skatulyázott ciklusos (block-nested loop) összekapcsolás
- Indexelt skatulyázott ciklusos összekapcsolás
- Összefésüléses rendező összekapcsolás
- Hasításos összekapcsolás

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

Legjobb eset, ha a kisebb reláció elfér a memóriában. Ezt használjuk belső relációnak. $B_R + B_S$ a költség.

**111. Skatulyázott (NestedLoop) összekapcsolásnál mennyi a költség legrosszabb esetben? (3 pont)**
Legrosszabb eset, ha mindkét relációból csak 1-1 lap fér bele a memóriába. Ilyenkor S-t minden R-beli rekordnál végig kell olvasni. Ilyenkor $N_R * B_S + B_R$ a költség.

**112. Blokk-Skatulyázott (BlockNestedLoop) összekapcsolásnál mennyi a költség legjobb esetben? (3 pont)**
A legjobb eset, ha a kisebb reláció elfér a memóriában. Ezt használjuk belső relációnak. $B_R + B_S$ a költség.

**113. Blokk-Skatulyázott (BlockNestedLoop) összekapcsolásnál mennyi a költség legrosszabb esetben? (3 pont)**
Legrosszabb eset, ha mindkét relációból csak 1-1 lap fér bele a memóriába. S-t minden R-beli lapnál végig kell olvasni. Ilyenkor $B_R * B_S + B_R$ a költség.

**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)**
Költsége: A rendezés költsége $+ B_S + B_R$

**116. Hasításos összekapcsolásnál mennyi a költség? (3 pont)**
Költsége: $2 * (B_R + B_S) + (B_R + B_S)$

**117. Hasításos összekapcsolásnál mekkora méretű kosarakat képezünk? (2 pont)**
Alkalmazzunk $h_1$-et az összekapcsolási mezőkre 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)**
Szorzódó valószínűségek. $N_R * [(s1/N_R) * (s2/N_R) * \ldots * ((sn/N_R)]$

**121. Hány sora van a $\sigma_{\theta_1 \vee \theta_2 v \ldots \vee \theta_n}(R)$ lekérdezés eredményének? (2 pont)**
$N_R * (1 - [(1 - s1/N_R) * (1 - s2/N_R) * \ldots * ((1 - sn/N_R)])$

**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)**
Maximális méret: $N_S$

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

**127. Mi a szabályos zárójelezések számának rekurzív képlete? (2 pont)**
$T(1) = 1$
$T(n) = \sum T(i) * T(n-i)$
$T(6) = 42$

**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 suboptimális sorrendjét milyen algoritmussal lehet előállítani, és a tartalmazási hálón milyen irányban halad a kiértékelés? (2 pont)**
Selinger Algoritmus: R1⋈R2⋈R2⋈R4. A kiértékelés alulról felfele 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)**
Összehasonlítottuk: Balról jobbra, balról jobbra és a memóriában összekapcsolva a harmadik táblával, valamint a középső ténytábla soraihoz kapcsolva a szélső dimenziótáblákat. Ezek közül a harmadik volt a legjobb.

**132. A Q(A,B) JOIN R(B,C) JOIN S(C,D) lekérdezésnek három kiértékelésénél milyen indexeket tételeztünk fel? (2 pont)**
Ugyanannyi soruk van: $T_Q = T_R = T_S = T$
Ugyanannyi helyet foglalnak: $B_Q = B_R = B_S = B$
A képméretek, vagyis az előforduló értékek száma azonos: $I_{Q.B} = I_{R.B} = I_{R.C} = I_{S.C} = I$

**133. Az R(A,B) JOIN S(B,C) lekérdezés eredményében mennyi a sorok száma? (2 pont)**
$T_{R \bowtie S} = T_R * T_S / I$

**134. Az R(A,B) JOIN S(B,C) lekérdezés eredménye hány blokkból áll? (2 pont)**
$(T_R * B_S + T_S * B_R)/I$

**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)**

| | |
|---|---|
| Az 1. join költsége | $B + T * B/I +$ |
| Az 1. join kiírása (output mérete) | $2 * T * B/I +$ |
| A 2. join költsége | $2 * T * B/I + [(T^2/I) * B]/I +$ |
| A teljes output kiírása összesen | $3 * T^2 * B/I^2$ |
| **Végeredmény** | $\boldsymbol{B + 5 * T * B/I + 4 * T^2 * B/I^2}$ |

**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)**
Megspórolhatjuk az 1. join eredményének kiírását majd újbóli beolvasását, vagyis $2 * (2 * T * B/I)$-t. Az eredmény ekkor: $\boldsymbol{B + T * B/I + 4 * T^2 * B/I^2}$

**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 összesen | $3 * T^2 * B/I^2$ |
| **Végeredmény** | $\boldsymbol{B + 2 * T * B/I + 3 * T^2 * B/I^2}$ |

**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)**
Ha a c/b arányt tekintjük, akkor azt mondhatjuk, hogy ez az arány 3/4-hez tart, ha T/I tart a végtelenbe. Vagyis ha T/I elég nagy, akkor a c költsége nagyjából 3/4-e a b-nek.

**139. A legjobb átfutás mit optimalizál? (2 pont)**
Legjobb átfutás: minden sort minél hamarabb. Először számoljon, aztán gyorsan térjen vissza

**140. A legjobb válaszidő mit optimalizál? (2 pont)**
Legjobb átfutás: minden sort minél hamarabb. Először számoljon, aztán gyorsan térjen vissza

**141. Adjuk meg a ROWID szerkezetét, és egy példát is rá Oracle esetében! (2 pont)**
<Blokk>.<Sor>.<Fájl>
Pl.: 00000006.0000.000X

**142. Mi az "Explain plan for<SQL-utasítás>" utasítás hatása? (2 pont)**
Elmenti a tervet (sorforrások + műveletek) Plan_Table-be

**143. Jellemezzük a SELECT *FROM empWHERE rowid= '00004F2A.00A2.000C' utasítást! (4 pont)**
- Egy sor megkeresése
- Azonnal a blokkra megy 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)**
Konzisztens állapot: kielégíti az összes feltételt (megszorítást)
Konzisztens adatbázis: konzisztens állapotú adatbázis

**145. Mit hívunk tranzakciónak és mi jellemző rá? (4 pont)**
Tranzakció: Konzisztenciát megtartó adatkezelő műveletek sorozata
Ezek után mindig feltesszük:
Ha T tranzakció konzisztens állapotból indul + T tranzakció csak egyedül futna le
=> T konzisztens állapotban hagyja az adatbázis

**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 megőrizze az adatbázis konzisztenciáját, azaz a tranzakció végrehajtása után is teljesüljenek az adatbázisban előírt konzisztenciamegszorítások (integritási megszorítások), azaz az adatelemekre és a közöttük lévő kapcsolatokra vonatkozó elvárások.

**148. Mit jelent a tranzakció elkülönítési tulajdonsága? (2 pont)**
Az a tény, hogy minden tranzakciónak látszólag úgy kell lefutnia, mintha ez alatt az idő alatt semmilyen másik tranzakciót sem hajtanánk végre.

**149. Mit jelent a tranzakció tartóssági tulajdonsága? (2 pont)**
Az a feltétel, hogy ha egyszer egy tranzakció befejeződött, akkor már soha többé nem veszhet el a tranzakciónak az adatbázison kifejtett hatása.

**150. A tranzakció-feldolgozónak milyen három feladata van? (3 pont)**
A tranzakciófeldolgozó a következő 3 feladatot hajtja végre:
- naplózás
- konkurenciavezérlés
- holtpont feloldása

**151. A tranzakciók melyik tulajdonságát biztosítja a naplózás? (1 pont)**
Annak érdekében, hogy a tartósságot biztosítani lehessen, az adatbázis minden változását külön feljegyezzük (naplózzuk) lemezen.

**152. A tranzakciók melyik tulajdonságát biztosítja a konkurenciakezelés? (1 pont)**
Ezek az eljárásmódok biztosítják azt, hogy teljesen mindegy, mikor történik a rendszerhiba vagy a rendszer összeomlása, a helyreállítás-kezelő meg fogja tudni vizsgálni a változások naplóját, és ez alapján vissza tudja állítani az adatbázist valamilyen konzisztens állapotába.

**153. Mi az ütemező feladata? (2 pont)**
Az ütemező (konkurenciavezérlés-kezelő) feladata, hogy meghatározza az összetett tranzakciók résztevékenységeinek egy olyan sorrendjét, amely biztosítja azt, hogy ha ebben a sorrendben hajtjuk végre a tranzakciók elemi tevékenységeit, akkor az összhatás megegyezik azzal, mintha a tranzakciókat tulajdonképpen 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. Felkészülés a hibára: naplózás
2. Hiba után helyreállítás: a napló segítségével egy konzisztens állapot helyreállítása

**156. Mit hívunk adatbáziselemnek? (2 pont)**
Az adatbáziselem (database element) a fizikai adatbázisban tártolt adatok egyfajta funkcionális egysége, amelynek 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; (D)
2. a pufferkezelő által használt virtuális vagy valós memóriaterület; (M)
3. a tranzakció memóriaterülete. (M)

**158. Mit jelent az INPUT(X) művelet? (2 pont)**
Az X adatbáziselemet tartalmazó lemezblokk másolása a memóriapufferbe.

**159. Mit jelent a READ(X,t) művelet? (4 pont)**
Az X adatbáziselem bemásolása a tranzakció t lokális változójába. Részletesebben: ha az X adatbáziselemet tartalmazó blokk nincs a memóriapufferben, akkor előbb végrehajtódik INPUT(X). Ezután kapja meg a t lokális változó X értékét.

**160. Mit jelent a Write(X,t) művelet? (4 pont)**
A t lokális változó tartalma az X adatbáziselem memóriapufferbeli tartalmába másolódik. Részletesebben: 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 a 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)**

Ha előbb az <END CKPT> naplóbejegyzéssel találkozunk, akkor 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)**

Ha a <START CKPT(T1,…,Tk)> naplóbejegyzéssel találkozunk előbb, az azt jelenti, hogy a katasztrófa az ellenőrzőpont-képzés közben fordult elő, ezért a T1,…,Tk tranzakciók nem fejeződtek be a hiba fellépéséig. 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 T 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. (2. és 3. egy lépésben történik.)
(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)**
R1. 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)**

**For every Ti with <Ti, commit> in log:**
- **For all <Ti, X, v> in log:**
  - **Write(X, v)**
  - **Output(X)**

**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. (Módosított REDO napló)

**173. Fogalmazzunk meg 3 különbséget az UNDO és REDO naplózás esetén! (3 pont)**
- Az adat változás utáni értékét jegyezzük fel a naplóba
- Máshová rakjuk a COMMIT-ot, a kiírás elé => megtelhet a puffer
- Az UNDO protokoll esetleg túl gyakran akar írni => itt el lehet halasztani az írást

**174. Mit hívunk piszkos puffereknek? (1 pont)**
Már végrehajtott, de lemezre még ki nem írt módosításokat tárol.

**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 megváltoztatását.

**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 | |

**Megjegyzés: A <T,COMMIT> naplóbejegyzés kiírása kerülhetett volna a 9) lépés elé vagy a 11) lépés mögé is.**

**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)**
Az UNDO naplózáshoz hasonlóan most is 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 nyomban lemezre kell írni, amint megjelenik a naplóban.

**183. Adjunk meg a működés közbeni ellenőrzőpont képzésének lépéseit UNDO/REDO naplózás esetén! (6 pont)**
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 vagy 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 naplóbejegyzések ideiglenesen az SGA (System Global Area) memóriapuffereiben tárolódnak, amelyeket a Log Writer (LGWR) háttérfolyamat folyamatosan ír ki lemezre. (Az SGA tartalmazza az adatbáziselemeket tároló puffereket is, amelyeket pedig a Database Writer háttérfolyamat ír lemezre.)

**189. Milyen problémát kell megoldania a konkurrencia-vezérlésnek? (4 pont)**
A tranzakciók közötti egymásra hatás az adatbázis-állapot inkonzisztenssé válását okozhatja, még akkor is, amikor a tranzakciók külön-külön megőrzik a konzisztenciát, és rendszerhiba sem történt.

**190. Mit hívunk ütemezőnek? (2 pont)**
A tranzakciós lépések szabályozásának feladatát az adatbázis-kezelő rendszer ütemező (scheduler) része végzi

**191. Mit hívunk ütemezésnek? (2 pont)**
Az ütemezés (schedule) 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, abortot rendel el, hogy a sorba- rendezhetőséget biztosítsa.

**193. Mit hívunk konfliktuspárnak? (2 pont)**
A konfliktus (conflict) vagy konfliktuspár 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) $r_i(X)$; $w_i(Y)$ konfliktus
(2) $w_i(X)$; $w_j(X)$ konfliktus
(3) $r_i(X)$; $w_j(X)$ és $wi(X)$; $r_j(X)$ is konfliktus.

**195. Mikor konfliktus-ekvivalens 2 ütemezés? (2 pont)**
Azt mondjuk, hogy két ütemezés konfliktusekvivalens (conflict-equivalent), ha szomszédos műveletek nem konfliktusos cseréinek sorozatával az egyiket átalakíthatjuk a másikká.

**196. Mikor konfliktus-sorbarendezhető egy ütemezés? (2 pont)**
Azt mondjuk, hogy egy ütemezés konfliktus-sorbarendezhető (conflict-serializable schedule), ha konfliktusekvivalens valamely soros ütemezéssel.

**197. Mi akonfliktus-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)**
Azt mondjuk, hogy egy ütemezés konfliktus-sorbarendezhető (conflict-serializable schedule), ha konfliktusekvivalens valamely soros ütemezéssel. A konfliktus-sorbarendezhetőség elégséges feltétel a sorbarendezhetőségre, vagyis egy konfliktussorbarendezhető ütemezés sorbarendezhető ütemezés is egyben.

**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)**
w1(Y); w2(Y); w2(X); w1(X); w3(X);
Intuíció alapján átgondolva annak, hogy T1 és T2 milyen értéket ír X-be, nincs hatása, ugyanis T3 felülírja X értékét. Emiatt S1 és S2 X-nek is és Y-nak is ugyanazt az értéket adja. Mivel S1 soros ütemezés, és S2-nek bármely adatbázis-állapotra ugyanaz a hatása, mint S1-nek, ezért S2 sorbarendezhető. Ugyanakkor mivel nem tudjuk felcserélni w1(X)-et w2(X)-szel, így cseréken keresztül nem lehet S2-t valamelyik soros ütemezéssé átalakítani. Tehát S2 sorba- rendezhető, de nem konfliktus-sorbarendezhető.

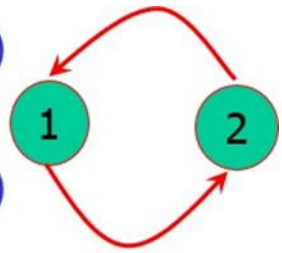
**201. Mi a konfliktus-sorbarendezhetőség tesztelésének alapötlete? (2 pont)**
Ha valahol konfliktusban álló műveletek szerepelnek S-ben, akkor az ezeket a műveleteket végrehajtó tranzakcióknak ugyanabban a sorrendben kell előfordulniuk a konfliktus-ekvivalens soros ütemezésekben, mint ahogyan az S-ben voltak.

**202. Mikor mondjuk, hogy egy S ütemezés alapján $T_1$*megelőzi* $T_2$-t?(5 pont)**
Adott a T1 és T2, esetleg további tranzakcióknak egy S ütemezése. Azt mondjuk, hogy T1 megelőzi T2-t, ha van a T1-ben olyan A1 művelet és a T2-ben olyan A2 művelet, hogy
- A1 megelőzi A2-t S-ben,
- A1 és A2 ugyanarra az adatbáziselemre vonatkoznak, és
- A1 és A2 közül legalább az egyik írás művelet.

**203. Adjuk meg egy S ütemezéshez tartozó megelőzési gráf definícióját! (5 pont)**
Ezeket a megelőzéseket a megelőzési gráfban (precedence graph) összegezhetjük. A megelőzési gráf csúcsai az S ütemezés tranzakciói. Ha a tranzakciókat Ti-vel jelöljük, akkor a Ti-nek megfelelő csúcsot az i egész jelöli. Az i csúcsból a j csúcsba akkor vezet irányított él, ha Ti <S $T_j$.

**204. Milyen kapcsolat van a konfliktus-ekvivalencia és a megelőzési gráfok között? (4 pont)**
S1, S2 konfliktusekvivalens => gráf(S1) = gráf(S2)
**Megjegyzés:**
**gráf($S_1$)=gráf($S_2$) $\not\Rightarrow$ $S_1$, $S_2$ konfliktusekvivalens**

**205. Adjunk példát arra, hogy két ütemezés megelőzési gráfja megegyezik, de nem konfliktus-ekvivalensek! (4 pont)**
**$S_1$=$w_1(A)$ $r_2(A)$ $w_2(B)$ $r_1(B)$**
**$S_2$=$r_2(A)$ $w_1(A)$ $r_1(B)$ $w_2(B)$**




**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,
- és 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.

- zárak
- időbélyegek
- érvényesítés

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

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

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

Tétel: Konzisztens, kétfázisú zárolású tranzakciók bármely S jogszerű ütemezését át lehet alakítani konfliktusekvivalens soros ütemezéssé.

Bizonyítás: S-ben részt vevő tranzakciók száma (n) szerinti indukcióval.

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

Csúcsai a tranzakciók és akkor van él Ti-ből Tj-be, ha $T_i$ vár egy olyan zár elengedésére, amit Tj 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)**

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

b. 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)**

Tegyük fel, hogy $T1,\ldots, T_n$ irányított kört alkot, ahol $T_i$ vár $T_{i+1}$-re az $A_i$ adatelem miatt. Ha mindegyik tranzakció betartotta, hogy egyre nagyobb indexű adatelemre kért zárat, akkor $A1 < A2 < A3 < An < A1$ áll fenn, ami ellentmondás. Tehát ez a protokoll is megelőzi a holtpontot, de itt is előre kell tudni, hogy milyen zárakat fog kérni egy tranzakció.

Még egy módszer, ami szintén optimista, mint az első: Időkorlát alkalmazása: ha egy tranzakció kezdete óta túl sok idő telt el, akkor ABORT-áljuk. Ehhez az kell, hogy ezt az időkorlátot jól tudjuk megválasztani.

**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.

**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, mivel Ugyanazok a meggondolások alkalmazhatók az osztott és kizárólagos zárakra is, mint korábban. Q.E.D.

**221. Osztott és kizárólagos zárak esetén adjuk meg a kompatibilitási mátrixot!(4 pont)**
Az osztott (S) és kizárólagos (X) zárak kompatibilitási mátrixa:

| Ha ilyen zár van már kiadva | Megkaphatjuk-e ezt a típusú zárat? S | X |
|---|---|---|
| S | igen | nem |
| X | nem | nem |

**222. Többmódú zárak kompatibilitási mátrixa segítségével hogyan definiáljuk a megelőzési gráfot? (5 pont)**
- a megelőzési gráf csúcsai a tranzakciók és akkor van él Ti-ből Tj-be, ha van olyan A adategység, amelyre az ütemezés során Zk zárat kért és kapott Ti, ezt elengedte, majd
- ezután A-ra legközelebb Tj kért és kapott olyan Zl zárat, hogy a mátrixban a Zk sor $Z_l$ oszlopában Nem áll.

**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 a következő ütemezést:

$$l_1(A);\ r_1(A);\ u_1(A);\ l_2(A);\ r_2(A);\ u_2(A);\ l_1(A);\ w_1(A);\ u_1(A);\ l_2(B);\ r_2(B);\ u_2(B)$$

Ha megnézzük az írás/olvasás műveleteket (r1(A); r2(A); w1(A); r2(B)), akkor látszik, hogy az ütemezés hatása azonos a T2T1 soros ütemezés hatásával, vagyis ez egy sorbarendezhető ütemezés zárak nélkül.
Rajzoljuk fel a megelőzési gráfot:


Mivel ez irányított kört tartalmaz, akkor ezt elvetnénk, mert nem lesz sorbarendezhető az az ütemezés, amiben már csak a zárak vannak benne.

**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)**
L2 erősebb L1-nél, ha a kompatibilitási mátrixban L2 sorában /oszlopában minden olyan pozícióban „NEM" áll, amelyben 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 |

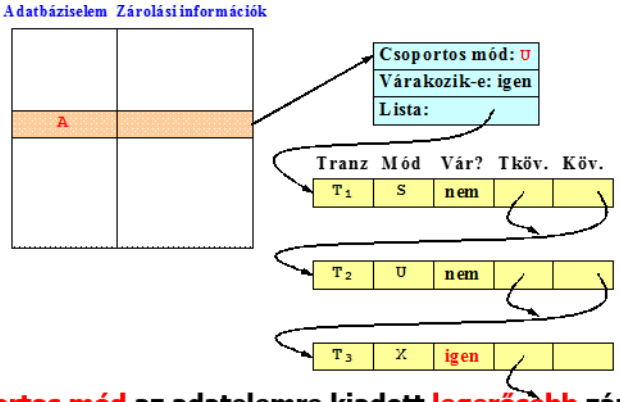
Az U módosítási zár úgy néz ki, mintha osztott zár lenne, amikor kérjük, és úgy néz ki, mintha kizárólagos zár lenne, amikor már megvan.

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

Az $inc_i$(X) művelet: a Ti tranzakció megnöveli az X adatbáziselemet valamely konstanssal.

| | 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)**




**Csoportos mód az adatelemre kiadott legerősebb zár:**

**a) S azt jelenti, hogy csak osztott zárak vannak;**

**b) U azt jelenti, hogy egy módosítási zár van, és lehet még egy vagy több osztott zár is;**

**c) X azt jelenti, hogy csak egy kizárólagos zár van, és semmilyen más zár nincs.**

**A *várakozási bit* (waiting bit) azt adja meg, hogy van-e legalább egy tranzakció, amely az A zárolására várakozik.**

**Az összes olyan tranzakciót leíró lista, amelyek vagy jelenleg zárolják A-t, vagy A zárolására várakoznak.**

**a) a zárolást fenntartó vagy a zárolásra váró tranzakció neve;**

**b) ennek a zárnak a módja;**

**c) a tranzakció fenntartja-e a zárat, vagy várakozik-e a zárra;**

**d) az adott tranzakció következő bejegyzése Tköv.**

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

1. Első beérkezett első kiszolgálása
2. Elsőbbségadás az osztott záraknak
3. Elsőbbségadás a felminősítésnek

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

- A kért 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)**

**Figyelmeztető zárak**

**SOR: Ha ilyen zár van már kiadva**

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

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

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

- IS<IX és S<X, de IX és S nem összehasonlítható (< csak parciális rendezés).
- A csoportos mód használatához vezessünk be egy SIX új zárat, (ami 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 \ Kérés | 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 T1 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 T2 tranzakció módosít vagy töröl a táblából néhány olyan sort, amely eleget tesz a lekérdezés feltételének.
- A T1 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)**

(A nem ismételhető olvasáshoz hasonlóan) ugyanez a helyzet akkor is, ha a T2 tranzakció beszúr olyan sorokat, amelyek eleget tesznek a lekérdezés feltételének. A lekérdezés másodszori futtatásakor most is más eredményt kapunk, mint az első alkalommal. Ennek az az oka, hogy most olyan sorokat is figyelembe kellett venni, amelyek az első futtatáskor még nem is léteztek. Az ilyen sorokat nevezzük fantomoknak (phantom)

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

A Ti tranzakció követi a faprotokollt, ha

1. Az első zárat bárhova elhelyezheti.
2. A későbbiekben azonban csak akkor kaphat zárat A-n, ha ekkor zárja van A apján.
3. Zárat bármikor fel lehet oldani (nem 2PL).
4. Nem lehet újrazárolni, azaz ha Ti elengedte egy A adategység zárját, akkor később nem kérhet rá újra (még akkor sem, ha A apján még megvan a zárja).

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

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

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

Időbélyegzés (timestamping):

- 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.

Érvényesítés (validation):

- Megvizsgáljuk a tranzakciók időbélyegzőit és az adatbáziselemeket, amikor a tranzakció véglegesítésre kerül. Ezt az eljárást a tranzakciók érvényesítésének nevezzük. Az a soros ütemezés, amely az érvényesítési idejük alapján rendezi a tranzakciókat, ekvivalens kell, hogy legyen az aktuális ütemezéssel.

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

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

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

Az Oracle utasítás-, valamint tranzakció szintű olvasási konzisztenciát biztosít. A kétféle olvasási konzisztencia eléréséhez az Oracle a rollback szegmensekben 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 jellemező a nem olvasásbiztos elkülönítési szintre a piszkos, fantom, nem ismételhető olvasásokra vonatkozóan? (3 pont)**

| | piszkos olvasás | nem ismételhető olvasás | fantomok olvasása |
|---|---|---|---|
| *nem olvasásbiztos* (read uncommitted) | lehetséges | lehetséges | lehetséges |

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

| | piszkos olvasás | nem ismételhető olvasás | fantomok olvasása |
|---|---|---|---|
| [redacted] | | | |
| *olvasásbiztos* (read committed) | nem lehetséges | lehetséges | lehetséges |

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

| | piszkos olvasás | nem ismételhető olvasás | fantomok olvasása |
|---|---|---|---|
| [redacted] | | | |
| *megismételhető olvasás* (repeatable read) | nem lehetséges | nem lehetséges | lehetséges |

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

| | piszkos olvasás | nem ismételhető olvasás | fantomok olvasása |
|---|---|---|---|
| [redacted] | | | |
| *sorbarendezhető* (serializable) | nem lehetséges | nem lehetséges | nem lehetséges |

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

DML-zárakat két szinten kaphatnak a tranzakciók:
- sorok szintjén
- és teljes táblák szintjén.

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

1. a kizárólagos (írási – X),
2. row share (RS) vagy subshare (SS),
3. row exclusive (RX) vagy subexclusive (SX),
4. share (S),
5. share row exclusive (SRX) vagy share-subexclusive (SSX)
6. és exclusive (X).