**Ellenőrző kérdések**

**4. Kis dolgozat kérdései**

**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 válaszidő: **az első sort** minél hamarabb
Számítás közben már térjen vissza (ha lehetséges)

**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 emp WHERE 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 tranzakciófeldolgozónak 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 a 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 | | | | | | |

**163. Adjunk 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