

Adatbázisok II

VI. Ellenőrzőkérdések

1. A konkurenciavezérlés

189. Milyen problémát kell megoldania a konkurencia-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.
- *MJ.: Akkor fordulhat elő ilyen, ha két, egyszerre futó tranzakció ugyan azt az adatrészt módosítja.*

190. Mit hívunk ütemezőnek? (2 pont)

- Az adatbázis-kezelő azon részét hívjuk ütemezőnek (scheduler), amely a tranzakciós lépések szabályozásának feladatát 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 sorbarendezhetőséget biztosítsa.

193. Mit hívunk konfliktuspárnak? (2 pont)

- A konfliktus (*conflict*) vagy konfliktuspár olyan egymást követő művelet-pár 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)

- Legyen T_i és T_j két különböző tranzakció ($i \neq j$).
 - a) $r_i(X); w_i(Y)$ konfliktus,
 - Mivel egyetlen tranzakción belül a műveletek sorrendje rögzített, és az adatbázis-kezelő ezt a sorrendet nem rendezheti át.
 - b) $w_i(X); w_j(X)$ konfliktus,
 - Mivel mind a kettőt ugyan azt az adatot módosítja. (Ha felcserélnénk őket, más lehet az eredmény.)
 - c) $r_i(X); w_j(X)$ és $w_i(X); r_j(X)$ is konfliktus.
 - Ha megcserélnénk a sorrendet, egyrészt az írások miatt más lenne az olvasott adat, másrészt a felcserélt írások sorrendje miatt X értékének a 4 művelet utáni változata is megváltozhat. (És ez a csere azt is jelentené, hogy tranzakción belüli sorrendet cserélünk fel.)

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.

- MJ.: Azaz nem konfliktusos cserékkel átvihető az egyik a másikba.

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

- ELV: nem konfliktusos cserékkel az ütemezést megpróbáljuk soros ütemezéssé átalakítani. Ha ezt meg tudjuk tenni, akkor az eredeti ütemezés sorbarendeázhető volt, ugyanis az adatbázis állapotára való hatása változatlan marad minden nem konfliktusos cserével.
- Megjegyzés: Két tranzakciónak két soros ütemezése van, az egyikben T_1 megelőzi T_2 -t, a másikban T_2 előzi meg T_1 -et.

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

- Azt mondjuk, hogy egy ütemezés konfliktus-sorbarendeázhető (*conflict-serializable schedule*), ha konfliktusekvivalens valamely soros ütemezéssel.
 - Azaz nem konfliktusos cserék által valamely soros ütemezést kaphatjuk.
- A konfliktus-sorbarendeázhetőség elégséges feltétel a sorbarendeázhetőségre, vagyis egy konfliktus-sorbarendeázhető ütemezés sorbarendeázhető 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)

- Azt állítjuk, hogy ez az ütemezés konfliktus-sorbarendeázhető. A következő cserékkel ez az ütemezés átalakítható a (T_1, T_2) soros ütemezéssé, ahol az összes T_1 -beli művelet megelőzi az összes T_2 -beli műveletet:

```

. r1 (A) ; w1 (A) ; r2 (A) ; w2 (A) ; r1 (B) ; w1 (B) ; r2 (B) ; w2 (B) ;
. r1 (A) ; w1 (A) ; r2 (A) ; r1 (B) ; w2 (A) ; w1 (B) ; r2 (B) ; w2 (B) ;
. r1 (A) ; w1 (A) ; r1 (B) ; r2 (A) ; w2 (A) ; w1 (B) ; r2 (B) ; w2 (B) ;
. r1 (A) ; w1 (A) ; r1 (B) ; r2 (A) ; w1 (B) ; w2 (A) ; r2 (B) ; w2 (B) ;
. r1 (A) ; w1 (A) ; r1 (B) ; w1 (B) ; r2 (A) ; w2 (A) ; r2 (B) ; w2 (B) ;

```

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

- $S_2: w_1(Y); w_2(Y); w_2(X); w_1(X); w_3(X);$

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

- Alapötlet: 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 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, és
 3. A_1 és A_2 közül legalább az egyik írás művelet.
- MJ.:
 - Másképpen fogalmazva: A_1 és A_2 konfliktuspárt alkotna, ha szomszédos műveletek lennének. Jelölése: $T_1 <_S T_2$.
 - Látható, hogy ezek pontosan azok a feltételek, amikor nem lehet felcserélni A_1 és A_2 sorrendjét. Tehát A_1 az A_2 előtt szerepel bármely S-sel konfliktusekvivalens ütemezésben. Ebből az következik, hogy ha ezek közül az ütemezések közül az egyik soros ütemezés, akkor abban T_1 -nek meg kell előznie T_2 -t.

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

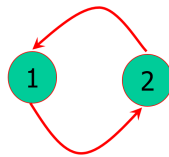
- Az utóbbi kérdésnél leírt megelőzéseket a megelőzési gráfban (*precedence graph*) összegezhethetjük. A megelőzési gráf csúcsai az S ütemezés tranzakciói. Ha a tranzakciókat T_i -vel jelöljük, akkor a T_i -nek megfelelő csúcsot az i egész jelöli. Az i csúcsból a j csúcsba akkor vezet irányított él, ha $T_i <_S T_j$.

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

- Lemma: S_1, S_2 konfliktusekvivalens $\implies \text{gráf}(S_1) = \text{gráf}(S_2)$
- Megjegyzés: $\text{gráf}(S_1) = \text{gráf}(S_2) \not\Rightarrow S_1, S_2$ konfliktusekvivalens

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

- Ellenpélda:
 - $S_1 = w_1(A) \ r_2(A) \ w_2(B) \ r_1(B)$
 - $S_2 = r_2(A) \ w_1(A) \ r_1(B) \ w_2(B)$



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 $a \rightarrow b$ élre az a csúcs megelőzi a b csúcsot a topologikus rendezésben.

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

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

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

- *MJ.*: A passzív ütemezés az ütemező egy eszköze a sorbarendehezhetőség elérésére.
- Passzív módszer:
 1. hagyjuk a rendszert működni,
 2. az ütemezésnek megfelelő gráfot tároljuk,
 3. egy idő után megnézzük, hogy van-e benne kör,
 4. 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)

- *MJ.*: Az aktív ütemezés az ütemező egy eszköze a sorbarendehezhetőség elérésére.
- Aktív módszer: az ütemező beavatkozik, és megakadályozza, hogy kör alakuljon ki.
- Az ütemezőnek több lehetősége is van arra, hogy kikényszerítse a sorbarendehezhető ütemezéseket:
 - záruk (ezen belül is még: protokoll elemek, pl. 2PL)
 - időbélyegek (time stamp)
 - érvényesítés
- *Fő elv*: inkább legyen szigorúbb és ne hagyjon lefutni egy olyan ütemezést, ami sorbarendehezhető, mint hogy fusson egy olyan, ami nem az.

2. Zárak használata

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

- *MJ*: A zárolási ütemező a konfliktus-sorbarendezhetőséget követeli meg, (ez erősebb követelmény, mint a sorbarendezhetőség).
- Tranzakciók konzisztenciája (consistency of transactions):
 - 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)

- Az ütemezések jogszerűsége (*legality of schedules*): 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)

- Ez az ütemezés jogszerű, de nem sorba rendezhető.

T₁	T₂	A	B
l₁(A) ; r₁(A) ;		25	
A := A+100 ;			
w₁(A) ; u₁(A) ;		125	
	l₂(A) ; r₂(A) ;	125	
	A := A*2 ;		
	w₂(A) ; u₂(A) ;	250	
	l₂(B) ; r₂(B) ;		25
	B := B*2 ;		
	w₂(B) ; u₂(B) ;		50
l₁(B) ; r₁(B) ;			50
B := B+100 ;			
w₁(B) ; u₁(B) ;			150

- Megjegyzés:
 - Kibővítjük a jelöléseinket a zárolás és a feloldás műveletekkel:
 - $l_i(X)$: a T_i tranzakció az X adatbáziselemre zárolást kér (lock).
 - $u_i(X)$: a T_i tranzakció az X adatbáziselem zárolását feloldja (unlock).
 - Konzisztencia: Ha egy T_i tranzakcióban van egy $r_i(X)$ vagy egy $w_i(X)$ művelet, akkor van korábban egy $l_i(X)$ művelet, és van később egy $u_i(X)$ művelet, de a zárolás és az írás/olvasás között nincs $u_i(X)$.
 - $T_i: \dots l_i(X) \dots r/w_i(X) \dots u_i(X) \dots$
 - Példa konzisztens tranzakciókra:

T₁: l₁(A) ; r₁(A) ; A := A+100 ; w₁(A) ; u₁(A) ;
l₁(B) ; r₁(B) ; B := B+100 ; w₁(B) ; u₁(B) ;

T₂: l₂(A) ; r₂(A) ; A := A*2 ; w₂(A) ; u₂(A) ;
l₂(B) ; r₂(B) ; B := B*2 ; w₂(B) ; u₂(B) ;

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

- A kétfázisú zárolás (*two-phase locking, 2PL*):
 - Szemléletesen:



- Példa:

$T_1: l_1(A); r_1(A); A := A+100; w_1(A); l_1(B);$
 $u_1(A); r_1(B); B := B+100; w_1(B); u_1(B);$
 $T_2: l_2(A); r_2(A); A := A*2; w_2(A); l_2(B);$
 $u_2(A); r_2(B); B := B*2; w_2(B); u_2(B);$

- Minden tranzakcióban minden zárolási művelet megelőzi az összes zárfeloldási műveletet. (Amíg a műveletei az adott adatot változtatják, addig nem engedi el azt.)

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.

2.1. A holtpont

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

- A holtpont: amikor egyik tranzakció sem folytatódhat, hanem örökké várakozniuk kell.

T_1	T_2	A	B
$(A); r_1(A);$		25	
	$l_2(B); r_2(B);$		25
$:= A+100;$			
	$B := B*2;$		
$(A);$		125	
	$w_2(B);$		50
$(B);$ elutasítva	$l_2(A);$ elutasítva		

- 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.
- Tétel: 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)

- 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. Ilyenkor biztos nem lesz holtpont, mert ha valaki megkap egy zárat, akkor le is tud futni, nem akad el. Az csak a baj ezzel, hogy előre kell mindent tudni.
- Feltesszük, hogy van egy sorrend az adategységeken és minden egyes tranzakció csak eszerint a sorrend szerint növekvően kérhet újabb zárat. Itt lehet, hogy lesz várakozás, de holtpont biztos nem lesz. Miért?

217. Mi a kiéheztes probléma és milyen megoldás van rá? (2 pont)

- Kiéhezés: többen várnak ugyanarra a zárra, de amikor felszabadul mindig elviszi valaki a tranzakció orra elől.
- Megoldás: adategységenként FIFO listában tartani a várakozókat, azaz mindig a legrégebben várakozónak adjuk oda a zárolási lehetőséget.

2.2. Osztott és kizárólagos zárok

Bevezetés:

- A legelterjedtebb zárolási séma két különböző zárat alkalmaz: az osztott zárat (*shared locks*) vagy olvasási zárat, és a kizárólagos zárat (*exclusive locks*) vagy írási zárat.
- Tetszőleges X adatbáziselemet vagy egyszer lehet zárolni kizárólagosan, vagy akárhányszor lehet zárolni osztottan, ha még nincs kizárólagosan zárolva.
- Amikor írni akarjuk X -et, akkor X -en kizárólagos zárral kell rendelkezünk, de ha csak olvasni akarjuk, akkor X -en akár osztott, akár kizárólagos zár megfelel.
- Feltételezzük, hogy ha olvasni akarjuk X -et, de írni nem, akkor előnyben részesítjük az osztott zárolást.
- Jelölések:
 - $sl_i(X)$: T_i tranzakció osztott zárat kér az X adatbáziselemre.
 - $xl_i(X)$: T_i kizárólagos zárat kér X -re.
 - $u_i(X)$: T_i felszabadítja X -et minden zár alól.

218. Osztott és kizárólagos zárok 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 zárok esetén mit hívunk az ütemezés jogszerűségének? (2 pont)

- Az ütemezések jogszerűsége: Egy elemet vagy egyetlen tranzakció zárol kizárólagosan, vagy több is zárolhatja osztottan, de a kettő egyszerre nem lehet.

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

- Konzisztens 2PL (2 fázisú) tranzakciók jogszerű ütemezése konfliktus-sorbarendezhető.

2.3. Kompatibilitási mátrixok

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

- Megjegyzés:
 - A kompatibilitási mátrix minden egyes zármódhoz rendelkezik egy-egy sorral és egy-egy oszloppal.
 - A sorok egy másik tranzakció által az X elemre elhelyezett zároknak, az oszlopok pedig az X -re kért zármódoknak felelnek meg.
 - A kompatibilitási mátrix használatának szabálya: Egy A adatbáziselemre C módú zárat akkor és csak akkor engedélyezhetünk, ha a táblázat minden olyan R sorára, amelyre más tranzakció már zárolta A -t R módban, a C oszlopban „igen” szerepel.
- Az osztott (S) és kizárólagos (X) zárok kompatibilitási mátrixa:

	S	X
S	igen	nem
X	nem	nem

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

- A megelőzési gráf csúcsai a tranzakciók és akkor van él T_i -ből T_j -be, ha van olyan A adategység, amelyre az ütemezés során Z_k zárat kért és kapott T_i ezt elengedte, majd
- Ezután A -ra legközelebb T_j kért és kapott olyan Z_l zárat, hogy a mátrixban a Z_k sor Z_l oszlopában *Nem* áll.

223. Többmódú záruk esetén a megelőzési gráf segítségével hogyan lehet eldönteni a sorbarendehezhetőséget? (3 pont)

- Tétel: Egy csak zárkéréseket és zárelengedéseket tartalmazó jogszerű ütemezés sorbarendehezhető 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 sorbarendehezhető 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 ($r_1(A); r_2(A); w_1(A); r_2(B)$), akkor látszik, hogy az ütemezés hatása azonos a T_2T_1 soros ütemezés hatásával, vagyis ez egy sorbarendehezhető ütemezés záruk nélkül.
- De ha felrajzoljuk a zárukra vonatkozó megelőzési gráfot (és ilyenkor persze nem nézzük, hogy milyen írások/olvasások vannak, hanem a legrosszabb esetre készülünk), akkor az irányított kört tartalmaz, akkor ezt elvetnénk, mert nem lesz sorbarendehezhető az az ütemezés, amiben már csak a záruk vannak benne.



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

- Tétel: 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 sorbarendehezhető.

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

- L_2 erősebb L_1 zárnál, ha a kompatibilitási mátrixban L_2 sorában /oszlopában minden olyan pozícióban „NEM” áll, amelyben L_1 sorában /oszlopában „NEM” áll.

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

- *Megjegyzés: Módosítási zár*
 - Az $ul_i(X)$ módosítási zár a T_i tranzakciónak csak X olvasására ad jogot, X írására nem. Később azonban csak a módosítási zárat lehet felminősíteni írásra, az olvasási zárat nem (azt csak módosításra).

- A módosítási zár tehát nem csak a holtponthelyzár problémát oldja meg, hanem a kiéheztetés problémáját is.
- Az U módosítási zár úgy néz ki, mintha osztozó zár lenne, amikor kérjük, és úgy néz ki, mintha kizárólagos zár lenne, amikor már megvan:

	S	X	U
S	igen	nem	igen
X	nem	nem	nem
U	nem	nem	nem

228. Mi az $\text{inc}_i(X)$ művelet és adjuk meg a növelési zár kompatibilitási mátrixát! (4 pont)

- Az $\text{inc}_i(X)$ művelet:
 - A Ti tranzakció megnöveli az X adatbáziselemet valamely konstanssal.
 - (Annak, hogy pontosan mennyi ez a konstans, nincs jelentősége.)
- MJ.: A műveletnek megfelelő növelési zárat (*increment lock*) $\text{il}_i(X)$ -szel jelöljük.
- Növelési zárok:

	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)

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

- Több különböző megközelítés lehetséges, mindegyiknek megvan a saját előnye:
 1. Első beérkezett első kiszolgálása (*first-come-first-served*):
Azt a zárolási kérést engedélyezzük, amelyik a legrégebb óta várakozik. Ez a stratégia azt biztosítja, hogy ne legyen kiéheztetés, vagyis a tranzakció ne várjon örökké egy zárra.
 2. Elsőbbségadás az osztozó zároknak (*priority to shared locks*):
Először az összes várakozó osztozó zárat engedélyezzük. Ezután egy módosítási zárolást engedélyezünk, ha várakozik ilyen. A kizárólagos zárolást csak akkor engedélyezzük, ha semmilyen más igény nem várakozik. Ez a stratégia csak akkor engedi a kiéheztetést, ha a tranzakció U vagy X zárolásra vár.
 3. Elsőbbségadás a felminősítésnek (*priority to upgrading*):
Ha van olyan U zárral rendelkező tranzakció, amely X zárra való felminősítésre vár, akkor ezt engedélyezzük előbb. Máskülönben a fent említett stratégiák valamelyikét követjük.

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

- *Megjegyzés: A figyelmeztető protokoll zárai (warning protocol):*
 - A közönséges zárok: S és X (osztozó és kizárólagos),
 - Figyelmeztető zárok: IS, IX (I=intention)
Például IS azt jelenti, hogy szándékunkban áll osztozó zárat kapni egy részelemezen.
- A kért zárnak megfelelő figyelmeztető zárokat 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 zárokra vonatkozó kompatibilitási mátrixot? (4 pont)

	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?
- Sor: Ha ilyen zár van már kiadva.

233. Hierarchikus adatok esetén miért vezetjük be az SIX zártípust és mi jellemző rá? (4 pont) 100. oldal 2. és 3. pont.

234. Adjuk meg a csoportos móddal kiegészített figyelmeztető zárokra vonatkozó kompatibilitási mátrixot! (5 pont) 102. oldal mátrixa, T igen, F nem.

235. Mit hívunk nem ismételtető olvasásnak és mi a probléma vele? (4 pont) 108. oldal első 4 pontja.

236. Mit hívunk fantom soroknak? (3 pont) 109. oldal teljesen. 237. Mikor követi egy tranzakció a faprotokollt? Adjuk meg a faprotokoll 4 szabályát! (4 pont) <http://people.inf.elte.hu/kiss/15ab2/konkurencia.ppt> 121. oldal tétel fölötti kék rész teljesen.

238. Hierarchiák, például B*-fa elemeinek zárolása esetén milyen feltétel adható az ütemezés sorbarendezhetőségére? (4 pont) 121. oldal tétel pontosan kimondva (az is kell, hogy jogszerű az ütemezés!).

239. Mi az időbélyegzési módszer lényege? Használunk-e ilyenkor záratokat? (4 pont) 131. oldal Időbélyegzés két alpontja pontosan. Nem használunk záratokat.

240. Adjunk meg három jellemzőt az Oracle konkurenciavezérlésére vonatkozóan! (3 pont) 174. oldal utolsó pontja.

241. Milyen olvasási konzisztenciát biztosít az Oracle és mivel éri ezt el? (3 pont) 175. oldal piros elnevezések és 176. oldal 1. pontja.

242. Adjuk meg az SQL92 ANSI/ISO szabványbanszereplő tranzakciós elkülönítési szinteket! (4 pont) 179. oldal táblázat első oszlopa.

243. Mi jellemző a nem olvasásbiztos elkülönítési szintre a piszkos, fantom, nem ismételtető olvasásokra vonatkozóan? (3 pont) 179. oldal táblázat első sora.

244. Mi jellemző az olvasásbiztos elkülönítési szintre a piszkos, fantom, nem ismételtető olvasásokra vonatkozóan? (3 pont) 179. oldal táblázat második sora.

245. Mi jellemző a megismételtető olvasás elkülönítési szintre a piszkos, fantom, nem ismételtető olvasásokra vonatkozóan? (3 pont) 179. oldal táblázat harmadik sora.

246. Mi jellemző a sorbarendezhető elkülönítési szintre a piszkos, fantom, nem ismételtető olvasásokra vonatkozóan? (3 pont) 179. oldal táblázat negyedik sora.

247. Milyen DML szintű záratokat használ az Oracle? (2 pont) 185. oldal középső pont két alpontjával.

248. Milyen zártípusokat használ az Oracle sorokra és táblákra? (6 pont) 185. oldal utolsó pont és 187. oldal utolsó előtti pont 5 alpontja.