Adatbázisok II

VI. Ellenőrzőkérdések

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.
- *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 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)

- Legyen T_i és T_j két különböző tranzakció ($i \neq j$).
  a) $r_i(X)$; $w_i(Y)$ konfliktus,
     - M ivel 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-sorbarendezhetőség elve? (3 pont)

– ELV: 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.
  • Azaz nem konfliktusos cserék által valamely soros ütemezést kaphatjuk.
– A konfliktus-sorbarendezhetőség elégséges feltétel a sorbarendezhetőségre, vagyis egy konfliktus-sorbarendezhető 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)

– Azt állítjuk, hogy ez az ütemezés konfliktus-sorbarendezhető. 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:

. $r_1(A)$; $w_1(A)$; $r_2(A)$; $\underline{w_2(A)}$; $\underline{r_1(B)}$; $w_1(B)$; $r_2(B)$; $w_2(B)$;
. $r_1(A)$; $w_1(A)$; $\underline{r_2(A)}$; $\underline{r_1(B)}$; $w_2(A)$; $w_1(B)$; $r_2(B)$; $w_2(B)$;
. $r_1(A)$; $w_1(A)$; $r_1(B)$; $r_2(A)$; $\underline{w_2(A)}$; $\underline{w_1(B)}$; $r_2(B)$; $w_2(B)$;
. $r_1(A)$; $w_1(A)$; $r_1(B)$; $\underline{r_2(A)}$; $\underline{w_1(B)}$; $w_2(A)$; $r_2(B)$; $w_2(B)$;
. $r_1(A)$; $w_1(A)$; $r_1(B)$; $w_1(B)$; $r_2(A)$; $w_2(A)$; $r_2(B)$; $w_2(B)$;

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

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

201. Mi a konfliktus-sorbarendezhetőség 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)

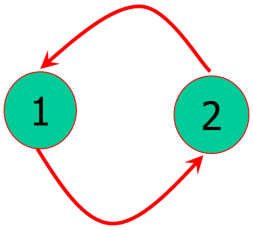
– Az utóbbi kérdésnél leírt megelőzéseket a megelőzési gráfban *(precedence graph)* összegezhetjük. A megelőzési gráf csúcsai az S ütemezés tranzakciói. Ha a tranzakciókat T_i-vel jelöljük, akkor a T_i-nek megfelelő csúcsot az i egész jelöli. Az i csúcsból a j csúcsba akkor vezet irányított él, ha $T_i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

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

- Lemma: S_1, S_2 konfliktusekvivalens $\implies$ gráf(S_1) = gráf(S_2)
- 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)

- 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 $a \to b$ 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)

- *MJ.: A passzív ütemezés az ütemező egy eszköze a sorbarendezhetőség 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 sorbarendezhetőség elérésére.*
- Aktív módszer: az ütemező beavatkozik, és megakadályozza, hogy kör alakuljon ki.
- Az ütemezőnek több lehetősége is van arra, hogy kikényszerítse a sorbarendezhető ütemezéseket:
  - zárak (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 sorbarendezhető, mint hogy fusson egy olyan, ami nem az.*

1. 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.

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

– 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: ... $l_i(X)$ ... $r/w_i(X)$ ... $u_i(X)$ ...

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

–