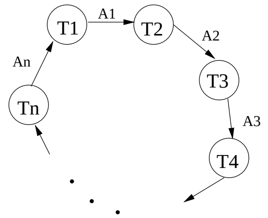
Adatbázisok elmélete 24. előadás

Csima Judit
Budapesti Műszaki és Gazdaságtudományi Egyetem
Számítástudományi Tsz.
I. B. 136/b
csima@cs.bme.hu

2003. Május 6.

Bizonyítás: Ha a sorosítási gráf nem DAG, akkor van benne kör:



A gráf felépítése miatt az ütemezésben biztos vannak $u_i(A_i)\dots l_{i+1}(A_i)$ szakaszok és ezek a szakaszok i szerint növőleg jönnek egymás után a 2PL miatt, hiszen $u_{i+1}(A_{i+1})$ csak $l_{i+1}(A_i)$ után jöhet.

Vagyis van egy
$u_1(A_1)\dots l_2(A_1)\dots u_2(A_2)\dots l_3(A_2)\dots\dots\dots u_n(A_n)\dots l_1(A_n)\dots$
rész az ütemezésben, de ez nem lehet, mert ekkor T_1 nem követi a 2PL-t.

Megjegyzések

- A 2PL előnye, hogy megelőzi a nem sorosítható ütemezés kialakulását, nem kell ABORT; hátránya viszont, hogy korlátozza a tranzakciókat
  Mindkét módszernek (gráf figyelése és ha kell, akkor ABORT, illetve 2PL) van előnye és hátránya is: a gráfos módszer nem óvatos feleslegesen, de az így előálló ABORT-ok miatt szükség van helyreállításra és a leállított tranzakciók feleslegesen dolgoztak; a 2PL-nél nincs ABORT és felesleges számolás, de a protokoll miatt esetleg túl sokáig tartjuk az zárakat, ami növeli a holtpont lehetőségét és lassítja a lefutást.
- Természetesen a 2PL csak elégséges feltétele a sorosíthatóságnak, de nem szükséges hozzá.
  Például az $l_1(A),\ u_1(A),\ l_2(A),\ u_2(A),\ l_1(B),\ u_1(B)$ ütemezés sorosítható, de ebben T_1 nem követi a 2PL-t.
- Az ütemező dolga az egyszerű tranzakció modellben: figyeli a zárkéréseket és
  - csak legális zárolásokat/ütemezéseket hagy futni (ehhez figyeli a zártáblát)
  - figyel a holtpontra (várakozási gráf, vagy protokoll-elemek)
  - figyel a sorosíthatóságra (sorosítási gráf vagy 2PL)

RLOCK/WLOCK modell

Láttuk, hogy amikor nem nézzük azt, hogy milyen írás és olvasási műveletek vannak, csak a zárkéréseket tekintjük, akkor túl szigorúak vagyunk. Baj még az is az egyszerű tranzakciómodellel, hogy akkor se hagyja, hogy két tranzakció egyszerre használjon valami adatot, ha nem is zavarnák egymást, mert pl. mindkettő olvasna csak. Ezekre a gondokra (részben) megoldás egy finomabb zármodell: **RLOCK/WLOCK** modell:

Két fajta zárkérés van (RLOCK és WLOCK), aszerint, hogy mit akar csinálni a tranzakció. Ezen kívül van még zárelengedés (UNLOCK).

A legális ütemezés jellemzői:

1. Az i-edik tranzakció, T_i, csak akkor írhatja az A adategységet, ha előtte írási zárat kért és kapott rá ($WLOCK_i(A)$) és a zárat még azóta nem engedte fel (nem volt még $UNLOCK_i(A)$). Az i-edik tranzakció, T_i, csak akkor olvashatja az A adategységet, ha előtte írási vagy olvasási zárat kért és kapott rá ($WLOCK_i(A)$ vagy $RLOCK_i(A)$) és

a zárat még azóta nem engedte fel (nem volt még $UNLOCK_i(A)$). Az lehetséges, hogy T_i először kér egy $RLOCK_i(A)$-t, majd aztán később (még az $UNLOCK_i(A)$ előtt) rákér egy $WLOCK_i(A)$-t is. A végső $UNLOCK_i(A)$ mindkét zárat feloldja.

2. Ha T_i zárolja az A adategységet, akkor később valamikor el is kell engednie a zárat ($RLOCK_i(A)$ és $WLOCK_i(A)$ után mindig van $UNLOCK_i(A)$).
3. Egyszerre több különböző tranzakciónak is lehet olvasási zárja ugyanazon az adategységen, de ha T_i-nek írási zárja van A-n, akkor másik tranzakció semmilyen zárat nem kaphat A-ra $UNLOCK_i(A)$ előtt.

Az ütemező egyik feladata ebben a zármodellben is az lesz, hogy csak legális zárkéréseket/ütemezéseket engedjen lefutni. Erre most is a várakoztatás lesz az eszköze, de most várhatóan kevesebb ilyen lesz, mert az egyszerre való olvasást nem tiltjuk.

Holtpont, éhezés RLOCK/WLOCK modellben

A holtpont és az éhezés ugyanazt jelenti ebben a modellben is, mint az egyszerű tranzakciómodellben és ugyanúgy lehet kezelni is. Az ütemező második feladata az lesz, hogy kezelje ezeket a problémákat valamelyik következő módszerrel:

1. Várakozási gráfot készít (pontosan ugyanolyan elv alapján, mint az egyszerű tranzakció modellben). Ebben a gráfban pontosan akkor van egy adott pillanatban irányított kör, ha az eddig a pillanatig beérkezett zárkérések holtpontot okoztak. (Bizonyitás ugyanúgy, mint az egyszerű tranzakciómodellnél). Ilyenkor ABORT valamelyik körbeli tranzakciónak.
2. Az egyszerű tranzakciómodellben látott holtpont-megelőző protokollok itt is mennek: növő sorrendben kérni zárat, minden zárat elkérni előre. (Bizonyitás ugyanúgy, mint az egyszerű tranzakciómodellnél).
3. Éhezés esetére: FIFO lista

Példa

Nézzük az alábbi írásokból és olvasásokból álló ütemezést:

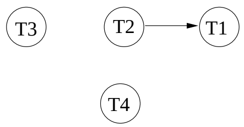
$$r_1(A),\ r_2(B),\ w_1(C),\ r_3(D),\ r_4(E),\ r_3(B),\ w_2(C),\ w_4(A),\ w_1(D)$$

RLOCK/WLOCK modellben vagyunk és tegyük fel, hogy a megfelelő (írási vagy olvasási) zárkérések mindig közvetlenül megelőzik a megfelelő műveletet, a zárelengedések pedig a tranzakciók végén, egyszerre történnek. Hogyan alakul a várakozási gráf ezen sorozat esetén? Lesz-e valamikor holtpont?

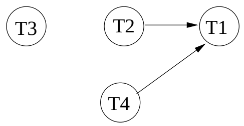
Az elején $rl_1(A),\ r_1(A),\ rl_2(B),\ r_2(B),\ wl_1(C),\ w_1(C),\ rl_3(D),\ r_3(D),\ rl_4(E),\ r_4(E)$ zárkérések és műveletek vannak, eddig még senki nem vár senkire.

Ezután $rl_3(B)$ jön $r_3(B)$ miatt, és T_3-nak most nem kell várnia T_2-re, mert két olvasási zár lehet ugyanott. Ezzel T_3 végzett is, jöhetnek a zárfeloldások: $u_3(D)$ és $u_3(B)$.

Ezután $wl_2(C)$ jön $w_2(C)$ miatt, de T_2-nek már várnia kell T_1-re:



Ezután $wl_4(A)$ jön $w_4(A)$ miatt, és T_4-nek is várnia kell T_1-re:



Végül $wl_1(D)$ jön $w_1(D)$ miatt, meg is kapja a zárat, T_1 befejeződik, így elengedi a zárait (a várakozási gráfban nem marad él) és ekkor T_2 és T_4 megkaphatja a kért zárakat, befejeződhet és elengedi az összes zárat, amit tartott.

Nem alakult ki holtpont semmikor, de várakozás volt.

Sorosíthatóság az RLOCK/WLOCK modellben

Hasonlóan az egyszerű tranzakciómodellhez, a sorosíthatóságról itt is pusztán a zárkérések alapján döntünk (ami azért most több infót ad arról, hogy milyen írások/olvasások történhetnek) és csak akkor akarunk egy zárkérésekből álló ütemezést sorosíthatónak minősíteni, ha függetlenül attól, hogy milyen legális írások és olvasások történnek, sorosítható.

Ennek eldöntésében itt is a **sorosítási gráf** segít: csúcsai a tranzakciók és akkor van él T_i-ből T_j-be, ha az ütemezésben vagy

- van olyan $u_i(A)$ $\ldots$ $wl_j(A)$ rész, ahol $u_i(A)$ csak egy olvasási zárat enged fel, és $u_i(A)$ és $wl_i(A)$ között A-ra senki se kér zárat vagy
- van olyan $u_i(A)$ $\ldots$ $rl_j(A)$ rész, ahol $u_i(A)$ egy írási zárat enged fel, és $u_i(A)$ és $rl_i(A)$ között A-ra senki se kér zárat vagy
- van olyan $u_i(A)$ $\ldots$ $wl_j(A)$ rész, ahol $u_i(A)$ egy írási zárat enged fel, és $u_i(A)$ és $wl_i(A)$ között A-ra senki se kér zárat.

Azaz majdnem minden olyan esetben kell él, amikor az egyszerű tranazakció modellben kellett, kivéve, ha olvasás után jön olvasás.

Ez azért van így, mert ebben a három esetben minden olyan soros ütemezésben, ami ekvivalens lehet a miénkkel, biztos, hogy T_j-nek T_i után kell jönnie. hiszen feltettük, hogy T_i is és T_j is bármi legálisat csinálhat A-val, amíg nála van a zár és két írás/olvasás művelet sorrendje csak akkor mindegy, ha mindkettő olvasás.

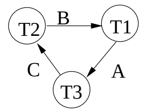
A sorosítási gráfra itt is igaz az egyszerű tranzakciómodellnél látott tétel:

Tétel. *Egy csak zárkéréseket és zárelengedéseket tartalmazó RLOCK/WLOCK modellbeli ütemezés pontosan akkor sorosítható, ha az előbbi módszerrel felrajzolt sorosítási gráf DAG*

Bizonyítás: Pontosan ugyanúgy megy, ahogyan az egyszerű tranzakciómodell esetén.

Példa

Az $rl_1(A),\ rl_2(B),\ rl_3(C),\ u_3(C),\ u_2(B),\ u_1(A),\ wl_1(B),\ u_1(B),\ wl_2(C),\ u_2(C),\ wl_3(A),\ u_3(A)$ RLOCK/WLOCK modellbeli ütemezéshez tartozó sorosítási gráf:

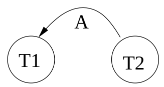


Ez nem DAG, így az ütemezés nem sorosítható.

Példa

Tekintsük az $rl_1(A),\ r_1(A),\ u_1(A),\ rl_2(A),\ r_2(A),\ u_2(A),\ wl_1(A),\ w_1(A),\ u_1(A),\ rl_2(B),\ r_2(B),\ u_2(B)$ ütemezést. (Ez az, aminek az egyszerű tranzakciómodellbeli megfelelője nem volt sorosítható).

Ha az RLOCK/WLOCK modellben rajzoljuk fel a sorosítási gráfot akkor



lesz a gráf, és ez DAG, ezért sorosítható lesz az ütemezés. Vagyis valóban nyertünk ezzel a modellel valamit: nem vagyunk annyira szigorúak feleslegesen, mint az egyszerű tranzakciómodellnél.

Sorosíthatóság kezelése az RLOCK/WLOCK modellben

Az ütemező harmadik dolga (a legális zárkérések kikényszerítése és a holtpontok kezelése mellett) itt is a sorosíthatóság elérése.

Eszközei ugyanazok, mint az egyszerű tranzakciómodellnél:

1. Figyeli a sorosítási gráfot és ha kör keletkezne, akkor az egyik körbeli tranzakciót ABORT-álja.
2. A **2PL** itt is működik: a T_i tranzakció $UNLOCK_i$ után nem kérhet se $RLOCK_i$-t, se $WLOCK_i$-t.

   Tétel. *Ha az RLOCK/WLOCK modellben, egy legális ütemezésben minden tranzakció követi a 2PL-t, akkor az ütemezéshez tartozó sorosítási gráf DAG, azaz az ütemezés sorosítható*

   Bizonyítás: Pontosan úgy, ahogy az egyszerű tranzakciómodellnél volt.

A két módszer előnyei/hátrányai pont ugyanazok, mint az egyszerű tranzakciómodellnél.