Tranzacții

Operații concurente asupra BD

- Un obiectiv major al SGBD este de a permite mai multor utilizatori să acceseze concurent datele partajate.
- BD este accesată simultan de mai mulţi utilizatori apar situaţii de conflict datorate accesului concurent la datele care constituie resursă comună.

Natura cererilor de acces la date

- Dacă cererile de acces sunt de tip regăsire, atunci secvenţialitatea accesului la mediul de memorare este suficientă şi nu mai este nevoie de precauţii suplimentare.
- Dacă unele cereri sunt de tip actualizare, este necesară aplicarea unor strategii adecvate de tratare a cererilor de acces.
  - Exemplu tipic: sistemele de rezervare a locurilor în care mai mulţi agenţi fac permanent modificări.
  - Pericolul: două procese, care citesc şi modifică valoarea aceluiaşi obiect ar putea interacţiona atunci când sunt executate simultan.
  - Rezultă necesitatea impunerii unor restricţii asupra execuţiei concurente a proceselor care fac operaţii de citire şi modificare a datelor.
  - O rezolvare imediată şi simplă ar fi blocarea BD pe durata rezolvării unei cereri. Dar asta echivalează cu blocarea concurenţei şi degradarea performanţelor sistemului, făcându-l uneori neutilizabil.

Algoritmii
de control ai concurenţei

- algoritmi de control prin blocare
- algoritmi de control prin marcare

Starea unei BD

- *Se spune ca o baza de date se afla intr-o **stare stabila** daca si numai daca in aceasta stare sunt rezolvate toate restrictiile de integritate impuse bazei de date*.
- *Se spune ca o baza de date se afla intr-o **stare instabila** daca in aceasta stare cel putin o restrictie de integritate nu este satisfacuta*.
- Prin prelucrarea datelor, baza de date trece din starea prezenta $\mathbf{S_p}$ intr-o stare urmatoare $\mathbf{S_u}$.

Regula de aur

- **Prelucrarile asupra bazei de date trebuie astfel realizate incat baza de date sa ajunga intotdeauna intr-o stare stabila**.
- Exista insa numeroase cauze care pot determina ca o baza de date sa evolueze dintr-o stare stabila intr-o stare instabila ceea ce evident nu poate fi acceptat.

Exemplu

- **E1**. Presupunem ca intr-o baza de date exista doua relatii **R1** si **R2** avand schemele **R1**(**A**, B, …) respectiv **R2**(**A**, C, …).
- In baza de date se impune urmatoarea regula de integritate:

  " Pentru orice valoare **a** din A, orice valoare **v** care se aduna la **B** trebuie sa fie scazuta din **C.** Rezultatul scaderii trebuie sa fie pozitiv. Daca rezultatul scaderii este negativ nu se accepta operatia de actualizare. "

Soluții

a) citeste B
B=B+v
scrie B
citeste C
C=C-v
Daca C≥0 scrie C.

b) citeste B
citeste C
B=B+v
C=C-v
scrie B
Daca C≥0 scrie C.

c) citeste C
C=C-v
Daca C≥0 (scrie C
citeste B
B=B-v
scrie B).

Anomalii ce pot sa apara

- Daca toate operatiile decurg normal, in final trebuie sa se ajunga pentru toate variantele intr-o stare $\mathbf{S_u}$ stabila adica o stare pentru care **V+C=constant**si **C≥0**.
- Evolutia dintr-o stare in alta se realizeaza in momentul scrierii intr-o relatie deoarece numai scrierea modifica valorile atributelor care definesc starea bazei de date.
- Se observa ca in fiecare caz in parte tranzitia implica trecerea printr-o stare instabila $\mathbf{S_i}$ in care numai o parte a regulii de integritate este respectata.
- Starile $\mathbf{S_p}$ si $\mathbf{S_u}$ sunt stari stabile.

Anomalii ce pot să apară

- Presupunem ca in fiecare caz, inaintea realizarii primei scrieri apare o cadere a sistemului. In acest caz continutul bazei de date nu este afectat si baza de date ramane in starea stabila $\mathbf{S_p}$.
- Presupunem ca sistemul cade dupa realizarea primei scrieri. In acest caz baza de date ramane instarea instabila $\mathbf{S_i}$ ceea ce reprezinta o eroare.
- Sistemul functioneaza corect dar C-v<0. Pentru variantele a) si b) acest lucru conduce la plasarea bazei de date in starea instabila Si, deci rezultatul este eronat. Pentru varianta c) baza de date ramane in starea stabila Sp ca si cum nu ar fi avut loc nici o operatie asupra bazei de date (operatiile de adunare si scadere se realizeaza in memoria tampon deci nu afecteaza continutul bazei de date).

- Din analiza efectuata se observa ca in cazul in care prelucrarea datelor implica mai multe operatii de actualizare conectate logic printr-o regula de actualizare a datelor, pot sa apara anomalii care conduc la plasarea bazei de date intr-o stare instabila ceea ce nu poate fi acceptat. Tocmai pentru a evita astfel de situatii a fost introdusa notiunea de ***tranzitie***.
- *Prin **tranzitie** se intelege un set de operatii de actualizare a continutului bazei de date, conectate logic printr-o regula de integritate, care ori se executa toate ori sunt anulate toate astfel incat in final baza de date se va afla intr-o stare stabila indiferent ce evinimente apar pe durata executarii acestor operatii.*

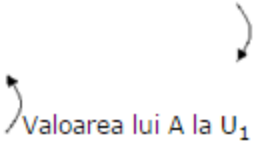
Exemplu

- **E2**. Se presupune ca asupra relatiei R1(A, …) se initializeaza simultan doua tranzactii T_1 si T_2 de la doi utilizatori diferiti, $\mathbf{U_1}$ si $\mathbf{U_2}$. Fiecare tranzactie trebuiesa citeasca aceeasi valoare din **A** si sa inlocuiasca vechea valoare cu valoarea A+10. Este evident ca rezultatul corect dupa realizarea celor doua tranzactii trebuie sa fie A+20.

Scenariul 1:

T_1		READ A	A:=A+10			**WRITE A**
T_2	READ A			A:=A+10	**WRITE A**	
Valoarea lui A la U_1	*	50	**60**	**60**	**60**	**60**
Valoarea lui A la U_1	50	50	**50**	**60**	**60**	*
Valoarea lui A in baza de date	50	50	**50**	**50**	**60**	**60**

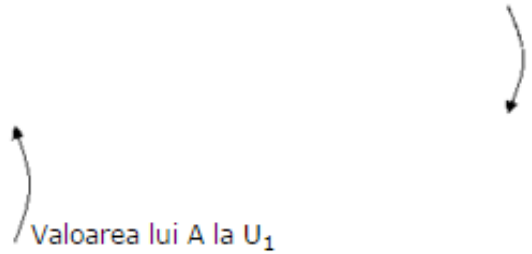
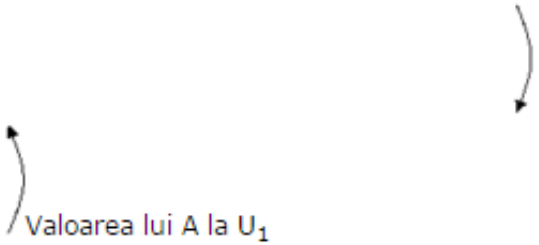
Scenariul 2:

T_1	READ A	A:=A+10	**WRITE A**			
T_2				READ A	A:=A+10	**WRITE A**
 Valoarea lui A la U_1	50	**60**	**60**	*	*	*
 Valoarea lui A la U_1	*	*	*	**60**	**70**	70
Valoarea lui A in baza de date	50	50	**60**	**60**	**60**	**70**

Exemplu

- E3. Se consideră tranzacția $\mathbf{T_1}$ care execută operația A:=A+1 pentru valoarea lui **A** din înregistrarea $\mathbf{I_1}$ și tranzacția $\mathbf{T_2}$ care execută aceeași operație asupra valorii lui **A** din înregistrarea $\mathbf{I_2}$.

Scenariu posibil de execuție

T_1	READ A			A:=A+1	WRITE A	
T_2		READ A	A:=A+1			WRITE A
	5	5	**5**	**6**	**6**	*
	*	8	**9**	**9**	**9**	**9**
Valoarea lui A (I_1)	5	5	**5**	**5**	**6**	**6**
Valoarea lui A (I_2)	8	8	**8**	**8**	**8**	**9**

- Iată două posibile tranzacţii care citesc variabila persistentă salariu într-o variabilă locală, o modifică şi apoi o pun la loc:

```
Begin T1;
1) x := Read(salariu);
2) x := x * 110/100;
3) Write(salariu, x);
End T1;
```

```
Begin T2;
1) y := Read(salariu);
2) y := y + 200;
3) Write(salariu, y);
End T2;
```

Scenariu posibil

`Begin T1;`	tt
`1) x := Read(salariu);`	tt
	ttBegin T2;
	tt1) y := Read(salariu);
`2) x := x * 110/100;`	tt
`3) Write(salariu, x);`	tt
	tt2) y := y + 200;
	tt3) Write(salariu, y);
`End T1;`	tt
	ttEnd T2;

Tranzacții

- O tranzacţie este o unitate singulară de execuţie care satisface două condiţii referitoare la BD:
  - BD este într-o stare coerentă înaintea şi după execuţia tranzacţiei.
  - BD poate fi într-o stare necoerentă în timpul execuţiei unei tranzacţii.

Rezultatele unei tranzacții

- O tranzacţie poate avea două rezultate:

1. dacă este completată cu succes, se spune că tranzacţia a fost efectuată iar BD ajunge într-o nouă stare coerentă;
2. dacă nu este executată cu succes, ea este abandonată iar BD trebuie refăcută în starea coerentă dinainte. O astfel de tranzacţie este rulată înapoi sau abandonată.

- O tranzacţie efectuată nu mai poate fi abandonată.

Delimitarea tranzacțiilor

- BEGIN TRANSACTION – începe o nouă tranzacţie
- COMMIT – salvează orice schimbări şi încheie tranzacţia curentă
- ROLLBACK – anulează orice schimbări făcute în timpul tranzacţei curente şi încheie tranzacţia

Proprietățile tranzacțiilor

- caracterul atomic: reprezintă proprietatea „tot sau nimic”. O tranzacţie este o unitate indivizibilă, care ori este efectuată în întregime, ori nu este efectuată de loc;
- coerenţa: o tranzacţie trebuie să transforme BD dintr-o stare coerentă în altă stare coerentă;
- izolarea: tranzacţiile sunt executate independent unele de altele. Efectele parţiale ale unei tranzacţii incomplete nu trebuie să fie vizibile pentru alte tranzacţii;
- durabilitatea: efectele unei tranzacţii încheiate cu succes sunt înregistrate definitiv în BD şi nu trebuie pierdute din cauza unei pene ulterioare.

Unitatea de acces

- O BD este partiţionată în mai multe unităţi de acces (items).
- Acestea sunt porţiuni ale BD care pot constitui obiectul unei operaţii de blocare (lock).
- Prin blocarea unei unităţi de acces, o tranzacţie poate împiedica accesul altor tranzacţii la unitatea blocată, până la momentul deblocării acestei unităţi de către tranzacţia care a efectuat blocarea.

Lock manager

- Gestiunea operaţiilor de blocare precum şi arbitrarea cererilor de blocare venite din partea tranzacţiilor este realizată de o componentă specială a SGBD, numită lock manager.

Natura şi dimensiunea unităţilor de acces

- Prin alegerea unităţilor de acces de mari dimensiuni, se reduce numărul operaţiilor de blocare, ceea ce înseamnă reducerea timpului consumat de sistem pentru gestiunea acestor operaţii, precum şi reducerea spaţiului de memorie necesar înregistrării blocajelor.
- Folosind unităţi de acces de mici dimensiuni, creşte gradul de concurenţă suportat de sistem, deoarece pot fi executate în paralel un număr mai mare de tranzacţii care operează în unităţi de acces diferite.

Natura şi dimensiunea unităţilor de acces

- În practică, dimensiunea potrivită a unităţilor de acces este dată de extinderea operaţiilor efectuate de tranzacţiile cu cea mai mare frecvenţă.
  - Astfel, dacă tranzacţia tipică presupune efectuarea unor operaţii de cuplare, atunci unitatea de acces va fi relaţia.
  - Dacă, majoritatea tranzacţiilor efectuează operaţii asupra unor tuple individuale, atunci va fi convenabil să se aleagă tupla ca unitate de acces.

Anomalii de interferenţă

- Interacţiunea necontrolată a două sau mai multe tranzacţii poate duce la apariţia unor stări inconsistente ale BD şi la producerea unor rezultate eronate.
- Două tranzacţii Ti şi Tj sunt susceptibile de **interferenţă dacă rezultatul execuţiei lor concurente poate fi diferit de rezultatul execuţiei seriale.**
- Între două tranzacţii poate să apară o interferenţă dacă acestea efectuează operaţii asupra unor date comune. Dacă aceste două tranzacţii sunt executate în mod concurent, atunci spunem că sunt conflictuale.
- Deci două tranzacţii Ti şi Tj sunt conflictuale dacă sunt concurente şi susceptibile de interferenţă.

Anomalii de interferenţă

- anomalia de actualizare pierdută
- anomalia de citire improprie
- anomalia de citire irepetabilă (citire murdară)

Anomalia de actualizare pierdută

- Corespunde unui conflict de tip scriere-scriere şi constă în faptul că rezultatul actualizării efectuate de o tranzacţie se pierde ca urmare a reactualizării aceleiaşi date de către o altă tranzacţie, fără ca reactualizarea să fie influienţată de rezultatul primei actualizări.

Exemplu:

- Fie următoarea execuţie concurentă a două tranzacţii T1 şi T2.

T1	T2
	Read A
Read A	
A = A + 5	
Write A	
	A = A + 10
	Write A

- În urma acestor tranzacţii valoarea lui A apare mărită cu 10, nu cu 15 aşa cum ar fi de aşteptat.
- Este ca şi cum tranzacţia T1 nici nu s-ar fi executat.

Anomalia de citire improprie

- Corespunde unui conflict de tip scriere-citire şi apare atunci când o tranzacţie surprinde o stare temporar inconsistentă a BD.

Exemplu

- fie următoarea execuţie concurentă a două tranzacţii T1 şi T2.

T1	T2
Read A	
A = A - 10	
Write A	
	Read A
	Read B
	C = A + B
	Write C
Read B	
B = B + 10	
Write B	

- În urma acestor tranzacţii valoarea sumei A + B este neschimbată. Intenţia era de a reţine în C valoarea acestei sume, dar datorită interferenţei, valoarea din C este cu 10 mai mică decât cea reală.

Anomalia de citire irepetabilă

- Corespunde unui conflict de tip citire-scriere şi apare atunci când aceeaşi tranzacţie găseşte valori diferite la citiri repetate ale aceleiaşi date.

Exemplu

- fie următoarea execuţie concurentă a două tranzacţii T1 şi T2.

T1	T2
Read A	
B = A	
Write B	
	Read A
	A = A + 10
	Write A
Read A	
C = A	
Write C	

- Se observă că, deşi valorile rezultate pentru B şi C ar trebui să fie egale în urma execuţiei tranzacţiei T1, ele sunt diferite din cauza interferenţei cu tranzacţia T2.

Primitivele LOCK şi UNLOCK

- Fie tranzacţiile T1 şi T2 execuţii diferite ale următoarei secvenţe de operaţii:

  Read A

  A = A + 1

  Write A

- unde A este o valoare existentă în baza de date.
- Fiecare din cele două tranzacţii citeşte valoarea lui A într-o zonă de lucru proprie, adună 1 la această valoare şi apoi scrie rezultatul în baza de date. După execuţia tranzacţiilor, este de aşteptat ca valoarea lui A să fie mărită cu 2.
- Totuşi, dacă tranzacţiile sunt executate concurent, este posibil ca rezultatul final să fie altul, funcţie de modul de interferenţă a tranzacţiilor.

Primitivele LOCK şi UNLOCK

- Cea mai simplă metodă de evitare a situaţiilor de genul celei prezentate, este de a permite accesul la valoarea lui A numai pentru o singură tranzacţie, pe toată durata executării tranzacţiei. Accesul celorlalte tranzacţii va fi temporar blocat.
- Acest lucru se poate realiza folosind două funcţii primitive LOCK (A) şi UNLOCK (A).
- Aceste funcţii se numesc primitive, deoarece secvenţa de operaţii corespunzătoare execuţiei lor nu poate fi întreruptă de alte operaţii.
- LOCK (A) şi UNLOCK (A) sunt operaţii indivizibile.

Primitivele LOCK şi UNLOCK

- Dacă o tranzacţie Tk execută cu succes o primitivă LOCK (A), atunci componenta lock manager a SGBD asigură accesul exclusiv al tranzacţiei Tk la valoarea A, interzicând accesul la această valoare a oricărei alte tranzacţii atâta timp cât tranzacţia Tk nu eliberează valoarea A prin execuţia primitivei UNLOCK (A).
- Se spune că valoarea A este blocată în acest interval de timp.
- O tranzacţie poate executa cu succes o primitivă LOCK() doar asupra unei valori care nu este blocată. În acest caz valoarea returnată de funcţia LOCK() este TRUE. Orice tentativă de a executa primitiva LOCK() asupra unei valori blocate va eşua, valoarea returnată fiind FALSE. Acesta este cel mai simplu mecanism de a asigura excluderea mutuală.

- Primitivele LOCK() şi UNLOCK() pot fi folosite pentru realizarea mecanismelor de sincronizare a tranzacţilor.
- Ordonarea secvenţială a paşilor a două sau mai multe tranzacţii care respectă regulile de mai sus, se spune că este legată.

Exemplu

- Pentru tranzacţiile T1 şi T2 considerate anterior, secvenţa de paşi folosind primitivele LOCK() şi UNLOCK() este următoarea:

  While NOT(LOCK(A))

  Read A

  A = A + 1

  Write A

  UNLOCK(A)

Probleme

- Rezultatul este corect, dar execuţia este pur secvenţială, nu este posibil nici un fel de paralelism între cele două tranzacţii.
- D.p.d.v. al timpului de execuţie, această situaţie este inacceptabilă şi de aceea se folosesc algoritmi care să relizeze ordonări secvenţiale legate, cu un grad de concurenţă a execuţiei cât mai ridicat, dar şi cu garanţia obţinerii de rezultate corecte.

Interblocarea

- Unul dintre principalele obiective ale oricărui sistem concurent este folosirea în comun a resurselor, adică partajarea acestora.
- În cazul bazelor de date concurente, cea ai importantă resursă partajabilă o constituie datele.
- Atunci când datele sunt partajate de către un grup de tranzacţii concurente şi fiecare tranzacţie deţine controlul exclusiv al unor date particulare, este posibil să se ajungă la situaţia în care unele tranzacţii nu-şi vor putea termina niciodată execuţia.

Exemplu

T1	T2
While NOT (LOCK(A))	While NOT (LOCK(B))
While NOT (LOCK(B))	While NOT (LOCK(A))
…	…
Prelucrare 1	Prelucrare 1
…	…
UNLOCK(A)	UNLOCK(B)
UNLOCK(B)	UNLOCK(A)

Interblocarea

- Presupunem că tranzacţiile T1 şi T2 îşi încep execuţia aproximativ în acelaşi moment. T1 cere şi obţine blocarea lui A iar T2 cere şi obţine blocarea lui B. Apoi T1 cere bocarea lui B, dar este pusă în aşteptare deoarece unitatea de acces B este bocată de tranzacţia T2. Concomitent, T2 cere blocarea lui A şi este pusă în aşteptare până când T1 deblochează pe A, dar T1 aşteaptă deblocarea lui B de către T2. În consecinţă, nici una dintre tranzacţii nu-şi poate continua execuţia, ambele fiind puse în aşteptare la nesfârşit. O astfel de situaţie se numeşte interblocare.

Metode de rezolvare a interblocării

1. metode de prevenire şi evitare a interblocării
2. metode de detecţie şi ieşire din interblocare

Condiţii pentru interblocare

- condiţia de **excludere mutuală** – tranzacţiile solicită controlul exclusiv al unităţilor de acces asupra cărora operează
- condiţia de **aşteptare pentru** – o tranzacţie care deţine controlul exclusiv asupra unor unităţi de acces este în aşteptare pentru altele
- condiţia de **completare** – nici o unitate de acces nu poate fi deblocată de către tranzacţia care o controlează înainte ca aceasta să termine toate operaţiile pe care le are de executat asupra unităţii respective
- condiţia de **aşteptare circulară** – există un lanţ circular de tranzacţii cu proprietatea că fiecare tranzacţie deţine controlul asupra unei unităţi de acces solicitată de următoarea tranzacţie din lanţ

Nesatisfacerea condițiilor

- **Negarea condiţiei 1** conduce la observaţia că nu poate apare interblocare în cazul tranzacţiilor care nu solicită accesul exclusiv la unităţile de acces. În cazul execuţiei concurente a unui set de tranzacţii care efectuează numai operaţii de citire nu poate apare interblocare.
- **Negarea condiţiei 2** conduce la o metodă de prevenire a interblocării care are la bază alocarea unităţilor de acces după criteriul „tot sau nimic”, numită metoda cererilor anticipate.
- **Negarea condiţiei 3** conduce la o metodă de detecţie şi ieşire din interblocare care presupune abandonarea (renunţarea) unora dintre tranzacţiile aflate în interblocare la un moment dat.
- **Negarea condiţiei 4** conduce la o metodă de prevenire a interblocării bazată pe ordonarea unităţilor de acces, numită metoda ordonării.

Metoda cererilor anticipate

- Blocarea unităţilor de acces de către tranzacţii se face după regula „tot sau nimic”. Fiecare tranzacţie emite deodată, toate cererile de blocare necesare execuţiei sale complete, în mod anticipat, înainte de a executa orice operaţie de actualizare.
- Sistemul acceptă sau respinge aceste cereri în bloc. Nu este posibil ca o tranzacţie să obţină blocarea unei părţi a unităţilor pe care doreşte să le acceseze, pentru ca pe parcurs să emită alte cereri d blocare.
- Astfel, nici o tranzacţie care a obţinut blocarea unor unităţi de acces nu va putea fi pusă în aşteptare.

Dezavantaje:

- tranzacţiile blochează unele dintre unităţile de acces pe o durată mai mare decât este necesar, ceea ce reduce nivelul de concurenţă al sistemului.
- favorizează apariţia fenomenului de amânare nedefinită sau „infometare” a tranzacţiilor. Tranzacţiile care solicită accesul la un număr mai mare de unităţi de acces ar putea fi menţinute în aşteptare un timp nedefinit, deoarece este puţin probabil ca resursele solicitate să se disponibilizeze toate în acelaşi moment. Aceste tranzacţii au şanse mult mai mici de a fi lansate în execuţie, faţă de tranzacţiile care solicită mai puţine resurse.
- există situaţii când această tehnică nu este aplicabilă. Este posibil ca pentru o tranzacţie care blochează două unităţi de acces să nu se poată preciza de la început care sunt acestea. Identificara celei de-a doua unităţi de acces poate să depindă de anumite valori din prima unitate de acces şi deci blocarea ei nu se poate face decât după ce s-a accesat prima unitate.

Metoda ordonării

- Metoda ordonării constă în stabilirea unei relaţii de ordine peste mulţimea unităţilor de acces. Tranzacţiile pot bloca unităţile de acces numai în această ordine prestabilită.
- Fie U1, U2, ..., Un unităţile de acces a căror ordonare este dată prin valoarea indicilor asociaţi.
- Presupunem că fiecare tranzacţie blochează unităţile de acces în ordinea crescătoare a indicilor.
- Dacă o tranzacţie Tx a blocat o unitate Ui, atunci T nu poate fi pusă în aşteptare decât pentru o unitate Uj, cu j>i. Dar o altă tranzacţie Ty care a blocat Uj nu poate fi în aşteptare pentru Ui, deoarece i < j.
- Deci interblocarea nu este posibilă.

Dezavantaje

- Afectează deasemenea nivelul de concurenţă al sistemului prin blocarea mai mult decât este necesar al unor unităţi de acces. Fie, de ex, o tranzacţie care doreşte accesul pentru o durată scurtă de timp la unitatea Ui şi un timp mai lung la unitatea Uj. Dacă i<j, atunci unitatea Ui va trbui să fie blocată pe toată durata blocării lui Uj.

Dezavantaje

- impune restricţii programatorilor în elaborarea tranzacţiilor. Cererile de acces la date trebuie să respecte ordinea impusă de sistem.
- În cazul BD complexe, realizarea unei ordonări a unităţilor de acces poate fi foarte dificilă (nu imposibilă), din cauza posibilităţilor foarte variate de divizare în unităţi de acces.
- Acestă metodă presupune existenţa apriorică a unei asemenea divizări, ceea ce exclue posibilitatea blocărilor

Marcarea timpului

- Nu este implicată nici o blocare, deci nu pot apare situaţii de impas (interblocare).
- Marca de timp este un identificator unic creat de SGBD, care indică timpul relativ de începere a unei noi tranzacţii.
- Mărcile de timp pot fi generate folosind ceasul sistemului sau prin declanşarea unui contor logic. Marcarea timpului este un protocol de control al concurenţei, în scopul ordonării globale a tranzacţiilor astfel încât, tranzacţiile mai vechi (cu mărci de timp mai mici) să aibă prioritate, în eventualitatea unui conflict.

Marcarea timpului

- Dacă o tranzacţie încearcă să citească sau să scrie o dată, aceste operaţii sunt permise numai dacă ultima reactualizare a respectivei date a fost efectuată de o tranzacţie mai veche.
- Altfel, tranzacţia care necisită operaţia de citire/scriere este reîncepută şi i se atribuie o nouă marcă de timp.
- Este necesar ca tranzacţiilor reîncepute să li se atribuie noi mărci de timp, pentru a evita să fie încontinuu abandonate şi reîncepute.

Regula de scriere a lui Thomas

- Pentru a obţine o concurenţă cât mai mare prin respingerea operaţiilor scoase din uz, se poate utiliza o modificare a protocolului de ordonare a mărcilor de timp:
  - Dacă tranzacţia T încearcă să scrie un articol x a cărui valoare a fost deja citită de către o tranzacţie mai nouă, tranzacţia T trebuie rulată înapoi şi reîncepută cu o nouă marcă de timp;
  - Dacă tranzacţia T încearcă să scrie un articol x a cărui valoare a fost deja scrisă de către o tranzacţie mai nouă, operaţia de scriere poate fi ignorată, pentru că valoarea pe care tranzacţia T vrea să o scrie se bazează pe o valoare veche a articolului x, valoare scoasă deja din uz.

Timpul	Operația	T1	T2	T3
t_1		Begin_trans		
t_2	Read x	Read x		
t_3	x=x+10	x=x+10		
t_4	Write x	Write x	Begin_trans	
t_5	Read y		Read y	
t_6	y=y+20		y=y+20	Begin_trans
t_7	Read y			Read y
t_8	Write y		Write y	
T_9	y=y+30			y=y+30
t_{10}	Write y			Write y
t_{11}	z=100			z=100
t_{12}	Write z			Write z
t_{13}	z=50	z=50		Commit
t_{14}	Write z	Write z	Begin_trans	
t_{15}	Read y	Commit	Read y	
t_{16}	y=y+20		y=y+20	
t_{17}	Write y		Write y	
t_{18}			Commit	

Regula de scriere a lui Thomas

- Să presupune că sunt trei tranzacţii concurente şi că mărcile lor de timp sunt la un anumit moment, în ordinea
  - $M_{T1} < M_{T2} < M_{T3}$
- La momentul t_8, operaţia de scriere a tranzacţiei T2 violează regula de scriere a mărcilor de timp şi drept urmare este abandonată şi reîncepută la momentul t_{14}.
- La momentul t_{14}, operaţia de scriere a tranzacţiei T1 poate fi ignorată utilizând regula de ignorare a scrierilor scoase din uz, deoarece ar fi fost suprascrisă prin operaţia de scriere a tranzacţiei T3 de la momentul t_{12}.