Controlul concurenței pe baza ordonării ştampilelor de tranzacții

Folosirea mecanismelor elementare de închidere, combinată cu protocolul de închidere în două faze (2PL) permite garantarea serializabilităşții execuțiilor de tranzacții. Ordinea tranzacțiilor în planificarea serială echivalentă este bazată pe ordinea în care tranzacțiile prelucrate închid obiectele de date la care trebuie să aibă acces. Ne amintim că în acest context, dacă o tranzacție solicită dreptul de a închide un obiect de date care este deja închis, ea poate fi "forțată" să aştepte până când accesul la obiect este eliberat.

O abordare diferită, care garantează de asemenea serializabilitatea, asociază cu tranzacțiile "ştampile de timp", în scopul ordonării acestora astfel îmcât să fie obținută o execuție echivalentă cu una serială.

Ştampila de timp este un identificator unic, creat de sistemul de gestiune a bazei de date, pentru a identifica o tranzacție. Vom nota cu TS(T) ştampila de timp a tranzacției T (TS provine de la originalul "time stamp"). De obicei, ştampilele de timp sunt allocate în ordinea în vcare tranzacțiile sosesc în sistemul bazei de date, ceea ce înseamnă că o ştampilă de timp ar putea fi asimilată cu momentul de intrare în lucru a unei tranzacții. Tehnicile de control al concurenței bazate pe ştampile de timp nu folosesc închideri, prin urmare nu pot să apară blocări.

Dintre metodele de generare a ştampilelor de timp menționăm:

- folosirea unui numărător, incrementat valoarea sa este alocată unei tranzacții. În această variantă, tranzacțiile sunt enumerate în secvența 1, 2, 3, etc. Dacă pentru un interval scurt de timp nu sunt executate tranzacții, conținutul numărătorului devine zero.
- Folosirea valorii curente a semnalului de tact al sistemului de calcul. În acest caz, generarea a două ştampile de timp în decursul aceluiaşi interval de tact trebuie evitată.

Un algoritm pentru ordonarea ştampilelor de timp

Ideea ce stă la baza algoritmului decurge din considerentele prezentate: tranzacțiile vor fi ordonate cu ajutorul ştampilelor de timp.Planificarea din care fac parte tranzacțiile va fi serializabilă, iar în execuția serială echivalentă tranzacțiile vor fi dispuse în ordinea valorilor ştampilelor de timp. Denumim această abordare "ordonare de tip ştampilă de timp", prescurtat (TO). Să subliniem deosebirea față de închiderea în două faze: în aceasta, o planificare este serializabilă dacă este echivalentă cu o execuție serială oarecare permisă de protocolul 2PL. În ordonarea cu ştampile de timp, planificarea este echivalentă cu o execuție serială particulară, care corespunde ordinei ştampilelor de timp ale tranzacțiilor.

În fapt, algoritmul trebuie să asigure condițiile în care pentru fiecare obiect de date la care în planificare au acces mai multe tranzacții, ordinea în care se realizează accesul nu încalcă serializabilitatea planificării. În vederea atingerii acestui obiectiv, algoritmul TO principal asociază cu fiecare obiect X al bazei de date două valori de ştampilă de timp:

1. read_TS (X), care este ştampila de timp pentru citire a obiectului de date X. Se defineşte ca fiind cea mai mare ştampilă de timp dintre toate valorile associate cu tranzacțiile care au reuşit să citească X cu succes.
2. write_TS (X)- este ştampila de timp pentru înregistrarea obiectului X. Se defineşte ca fiind cea mai mare ştampilă de timp dintre toate valorile de ştampilă associate cu tranzacțiile ce au înregistrat obiectul X cu succes.

Să analizăm evenimentele care se produc în decursul evoluției algoritmului. Oricând o tranzacție oarecare T încearcă să inițieze o operațiune de citire pe X, Read X, sau o operațiune de înregistrare Write X, algoritmul TO principal compară ştampila de timp a tranzacției T cu valorile ştampilei de timp pentru citire sau înregistrare ale obiectului X, pentru a verifica dacă nu este cumva încălcată ordinea ştampilelor de timp ale planificării tranzacțiilor. Dacă operațiunea inițiată încalcă ordinea, tranzacția T nu respectă ordonarea planificării seriale echivalente şi, ca urmare, T va fi abandonată. În continuare, tranzacția T este din nou prezentată sistemului de gestiune a bazei de date, ca o tranzacție nouă, cu o ştampilă de timp de valoare nouă.. În ipoteza că tranzacția T a fost abandonată şi apoi reintrodusă în sistem, orice altă tranzacție T1 care eventual a folosit o valoare înregistrată de către T trebuie să fie, de asemenea, reintrodusă în sistem. Similar, orice tranzacție T2 care a folosit, eventual, o valoare înregistrată de T1, trebuie reintrodusă, ş.a.m.d. Procesul descries poartă numele de "reintroducere în cascadă" şi constitue una dintre dificultățile associate cu algoritmul TO principal, deoarece execuşiile produse nu pot fi recuperate.

Respectarea ordonării tranzacțiilor după valorile ştampilelor de timp trebuie verificată de algoritmul de control al concurenței în următoarele două cazuri:

1. O tranzacție T, a cărei ştampilă de timp are valoarea TS (T), inițiază o operație Write X. Distingem situațiile prezentate mai jos:
    a. Dacă read_TS (X) > TS (T), sau write_TS(X) > TS(T), atunci T este abandonată, "reîntoarsă", iar operația inițiată respinsă. Această evoluție are loc deoarece o tranzacție cu ştampila de timp superioară față de TS (T)- şi, prin urmare, amplasată în urma tranzacției T în ordonarea ştamplielor de timp- a reuşit deja să citească sau să înregistreze valoarea obiectului X înainte ca tranzacția T să înregistreze acest obiect, încălcând astfel ordinea ştampilelor de timp.

b. Dacă nu este îndeplinită condiția specificată la punctual "a" de mai sus, operația Write X este executată, iar write_TS (X) ia valoarea TS (T).

2. Tranzacția T inițiază o operație Read X. În această situație distingem posibilitățile descrise în continuare.
   a. Dacă write_TS (X) > TS(T), atunci tranzacția T este abandonată, "reîntoarsă", iar operația de citire este respinsă. Evoluția descrisă se produce deoarece o tranzacție oarecare cu ştampila de timp mai mare decât TS (T)- amplasată în urma tranazacției T în ordonarea ştampilelor de timp- a reuşit deja să înregistreze valoarea obiectului de date X, înainte ca tranzacția T să fi citit X.
   b. Dacă write_TS (X) ≤ TS (T), atunci operațiunea Read X din tranzacția T este executată, iar valoarea read_TS (X) devine egală cu cea mai mare dintre valorile TS (T) sau read_TS (X) curentă.

Rezultă aşadar, că algoritmul TO principal verifică apariția într-o ordine incorectă a două operațiuni conflictuale şi o respinge pe ultima dintre acestea, abandonând tranzacția care a inițiat-o. Prin urmare, execuția produsă de algoritmul TO principal este cu siguranță "serializabilă la conflict".

Exemplu: Fie o planificare a tranzacțiilor T1, T2, T3, care au acces la obiectele A, B, C ale unei baze de date. În figura care urmează, sunt indicate ştampilele de timp ale tranzacțiilor şi duratele de citire şi înregistrare ale obiectelor de date A, B, C.

T1	T2	T3	A	B	C
200	150	175	RT=0	RT=0	RT=0
			WT=0	WT=0	WT=0
Read B				RT=200	
	Read A		RT=150		
		Read C			RT=175
Write B				WT=200	
Write A			WT=200		
	Write C				
	Abort				
		Write A			

După cum se vede, la început presupunem că la început fiecare din obiectele de date are atât timpul de citire, cât şi timpul de înregistrare de valoare 0. Ştampilele de timp ale tranzacțiilor sunt obținute atunci când acestea notifică planificatorul că încep să „lucreze", deci să fie executate. Observăm că tranzacția T1 execută primul acces la date, deşi nu are cea mai mică valoare a ştampilei de timp. De

fapt, tranzacția T2 era prima care ar fi trebuit să notifice planificatorul că începe execuția, urmtă de T3, iar T1 ar fi trebuit să fie ultima tranzacţie ce începe execuția.

În decursul primului pas al planificării propuse, prima acțiune realizează citirea obiectului B de către tranzacția T1. Deoarece timpul de înregistrare pentru B este mai mic decât

ştampila de timp a tranzacției T1, operațiunea de citire este fizic realizabilă şi acestei tranzacții i se permmite începerea execuției. Timpului de citire al obiectului B i se atribue valoarea 200, care este ştampila de timp a tranzacției T1. În mod asemănător, a doua şi a treia acțiune de citire sunt legale, astfel încât timpul de citire al fiecărui obiect de date ia valoarea ştampilei de timp a tranzacției care efectuează citirea.

În al patrulea pas al planificării, T1 înregistează B. Deoarece timpul de citire a obiectului de date B nu este mai mare decât ştampila de timp a tranzacției T1, înregistrarea este fizic realizabilă. Totodată, întrucât timpul de înregistrare a obiectului B nu este mai mare decât ştampila de timp a tranzacției T1, trebuie să fie realizată efectiv operația de înregistrare. Odată cu efectuarea acesteia, timpul de înregistrare a obiectului B ia valoarea 200, adică valoarea ştampilei de timp a tranzacției T1, care face înregistrarea.

În pasul al cincilea, procesul evoluează în manieră asemănătoare, cu operația de înregistrare a obiectului A de către T1 şi cu atribuirea valorii ştampilei de timp a tranzacției T1 către timpul de înregistrare a obiectului A.

În decursul următorului pas, al şaselea, tranzacția T2 încearcă să înregistreze obiectul de date C. Remarcăm însă că că C a fost citit anterior de către T3, tranzacție ce a fost executată teoretic la momentul 175 (valoarea ştamplei sale de timp), pe când T2 ar încerca să înregistreze C la momentul 150, care este valoarea ştampilei de timp a acestei tranzacții. Acțiunea încercată de T2 conduce la un comportament imposibil din punct de vedere fizic, aşadar această tranzacție trebuie re- întoarsă („rolled back”), fiind în final abandonată.

Ultimul pas al planificării este reprezentat de înregistrarea obiectului de date A de către T3. Deoarece timpul de citire a obiectului A are valoarea 150, mai mică decât ştampila de timp a tranzacției T3, de valoare 175, înregistrarea este legală. Observăm însă că în obiectul A a fost înregistrată valoarea 200 de către tranzacția T1-fapt petrecut teoretic la momentul 200, care este valoarea ştampilei de timp a tranzacției T1. În consecință, tranzacția T3 nu va fi returnată, dar nici nu va înregistra informație în obiectul A.