11.TRANZACŢII ŞI ACCES CONCURENT

Asa cum s-a exemplificat in primul capitol, atunci cand mai multe programe opereaza simultan pe aceleasi date pot sa apara situatii in care continutul bazei de date devine inconsistent: daca pasii aceluiasi program de rezervare de locuri rulat de la doua agentii de voiaj diferite sunt ca in tabelul de mai jos, desi se rezerva doua locuri numarul de locuri disponibile scade cu doar o unitate

Moment de timp	Agentia 1	Agentia 2	A în BD
t1	READ A		10
t2		READ A	10
t3	A = A – 1		10
t4		A = A - 1	10
t5	WRITE A		9
t6		WRITE A	9

In cazul accesului la aceleasi date, ca mai sus, se spune ca executiile celor doua programe sunt ***concurente*** sau ca exista un ***acces concurent la date***. Scopul acestui capitol este de a studia modalitatile de evitare a inconsistentelor precum si a problemelor ridicate de mecanismele folosite pentru aceasta.

6.1. Terminologie folosita

In acest prim paragraf sunt definiti principalii termeni folositi ca punct de plecare in studierea tranzactiilor si a accesului concurent la date. Pe parcurs vor fi introduse si alte notiuni care deriva din acestea:

a. Tranzactie

Notiunea de tranzactie va fi rafinata in paragrafele urmatoare. Definitia urmatoare este doar una de lucru pentru intelegerea celorlalti termeni din acest paragraf.

Definitie: O tranzactie este o singura executie a unui program.

In exemplul anterior exista doua tranzactii, T1 si T2 care erau doua executii ale aceluiasi program: T1 executia programului de rezervare lansata de Agentia 1 iar T2 este cea de la Agentia 2. Nu inseamna insa ca un set de tranzactii care opereaza simultan pe o baza de

date trebuie sa contina doar executii ale aceluiasi program. Putem avea de exemplu o tranzactie care rezerva un loc si o alta care anuleaza o rezervare de loc, ca in exemplul urmator:

Moment de timp	Tranzactia 1 lansata de Agentia 1: rezervare loc	Tranzactia 2 lansata de Agentia 2: anulare rezervare	A în BD
t1	READ A		10
t2		READ A	10
t3	A = A – 1		10
t4		A = A +1	10
t5	WRITE A		9
t6		WRITE A	11

Si in acest caz rezultatul este o inconsistenta a bazei de date deoarece numarul de locuri disponibile trebuia sa ramana constant.
Pe o aceeasi baza de date pot rula simultan mai multe tranzactii care lucreaza fiecare nu cu un singur element din baza de date (A din exemplul anterior) ci cu mai multe: fiecare tranzactie poate citi mai multe date si poate sa si scrie mai multe date in baza de date (nu neaparat cele citite ci si altele).
Operatiile efectuate de tranzactii care nu sunt de scriere sau de citire de date din baza de date nu duc la inconsistente: de exemplu nu incrementarea sau decrementarea lui A din exemplul anterior sunt cauza inconsistentelor ci ordinea in care s-au facut scrierile rezultatelor in baza de date. De aceea in unele dintre exemplele din acest capitol nu vom mai figura acest tip de operatii. Iata o executie concurenta pentru patru tranzactii:

Timp	T1	T2	T3	T4
t1	READ A			
t2	READ B			
t3		READ A		
t4		READ B		
t5			WRITE B	
t6		WRITE B		
t7				READ B
t8				READ C
T9	WRITE A			
t10		WRITE C		

Se observa ca putem avea:

- tranzactii scriu date care anterior au fost citite (ca T1 il scrie pe A, citit anterior),
- tranzactii care scriu date care nu au fost anterior citite, calculate eventual pe baza altora citite din baza de date (T2 scrie pe C pe care nu l-a citit, dar a citit A si B)
- tranzactii care doar citesc date fara sa scrie (T4)
- tranzactii care doar scriu date fara sa citeasca anterior ceva din baza de date (T3)

b. Articol (al unei baze de date)

Definitie: Un articol este o portiune a bazei de date care se poate citi sau scrie sau bloca/debloca printr-o singura operatie de READ, WRITE, LOCK respectiv UNLOCK.

In exemplul anterior am folosit articolele simbolice A, B si C. In cazurile reale un articol poate fi:

- O intreaga tabela
- O linie (sau o multime de linii) dintr-o tabela
- O celula dintr-o tabela (valoarea unei coloane de pe o linie a tabelei)
- Orice alta portiune a bazei de date care indeplineste conditia din definitie in concordanta cu facilitatile puse la dispozitie de SGBD-ul respectiv.

Exemplu: sistemul Oracle blocheaza automat orice linie modificata de o comanda de tip UPDATE pana cand tranzactia care a efectuat operatie fie ***comite*** modificarile (le face permanente in baza de date) fie le ***revoca***. In acest caz articolele sunt deci linii ale tabelei actualizate.

c. Planificare

Definitie: O planificare reprezinta ordinea in care sunt executati de SGBD pasii elementari ai unui set de tranzactii.

Planificarea este deci o lista de pasi pentru un set de tranzactii care se executa concurent si arata ca SGBD-ul executa acesti pasi in exact acea ordine.
In acest capitol vom reprezenta doar pasii care semnifica o interactiune a tranzactiei cu datele din baza de date:

- READ – citirea unui articol
- WRITE – scrierea unui articol
- LOCK (in diversele sale forme) – blocarea unui articol
- UNLOCK – deblocarea unui articol

Pentru cazurile in care operatiile de scriere nu devin permanente in baza de date decat dupa comitere putem avea si pasi de tipurile urmatoare:

- COMMIT – comiterea modificarilor efectuate de o tranzactie
- ROLLBACK – revocarea modificarilor efectuate de o tranzactie

Asa cum s-a mentionat, celelalte operatii nu ridica probleme de acces concurent. Exista mai multe moduri de a reprezenta o planificare:

- Sub forma tabelara, ca in exemplele anterioare de executie concurenta. Coloana "Timp" poate lipsi, ordinea executiei este de sus in jos.
- Sub forma unei liste.

Pentru a doua forma sa consideram urmatoarele notatii:

- $R_i(A)$ - semnifica: Tranzactia T_i citeste articolul A
- $W_i(A)$ - semnifica: Tranzactia T_i scrie articolul A

In acest caz ultima planificare (pentru T1, T2 si T3) se mai poate scrie si astfel:

$R_1(A); R_1(B); R_2(A); R_2(B); W_3(B); W_2(B); R_4(B); R_4(C); W_1(A); W_1(C)$

d. Planificare seriala

Definitie: O planificare in care pasii fiecarei tranzactii sunt succesivi, fara sa fie intercalati pasi ai altor tranzactii se numeste planificare seriala.

Exemplu de planificare seriala pentru doua tranzactii T1 si T2:

$R_1(A); R_1(B); R_1(C); W_1(B); R_2(B); R_2(C); W_2(A); W_2(C)$

Pasii lui T1 Pasii lui T2

Planificarile seriale nu ridica probleme de consistenta (sunt planificari "bune" din punct de vedere al executiei concurente). De aceea unul din obiectivele acestui capitol este acela de a gasi planificari care sa se comporte la fel cu o planificare seriala.

e. Blocarea articolelor

Cum s-a mentionat si in primul capitol, una dintre metodele de a obtine planificari care sa nu ridice probleme privind consistenta datelor dupa executia tranzactiilor este aceea a blocarii articolelor.

Definitie: Blocarea unui articol de catre o tranzactie semnifica faptul ca acea tranzactie obtine din partea sistemului (SGBD) anumite drepturi speciale de acces care impiedica alte tranzactii sa efectueze anumite operatii asupra acelui articol.

Exista doua categorii de blocari:

- Blocari exclusive: celelalte tranzactii nu pot sa execute operatii asupra articolului blocat. Aceste blocari sunt denumite in literatura de specialitate si Exclusive Locks sau Write Locks.
- Blocari partajate: celelalte tranzactii pot sa execute doar anumite tipuri de operatii asupra articolului blocat. Aceste blocari sunt denumite in literatura de specialitate si Shared Locks sau Read Locks

Exemplu:

- Pentru a scrie un articol, o tranzactie trebuie sa obtina anterior un Write Lock asupra acestuia: nici o alta tranzactie nu poate citi sau scrie acel articol pana cand el nu este deblocat
- Pentru a citi un articol o tranzactie trebuie sa obtina anterior un Read Lock asupra lui. Mai multe tranzactii pot sa blocheze acelasi articol pentru citire insa nici o tranzactie nu il poate scrie. O tranzactie poate sa obtina un Write Lock pe articolul respectiv abia dupa deblocarea acestuia de catre ***toate*** tranzactiile care l-au blocat pentru citire.

Articolele pot fi deblocate unul cate unul de catre tranzactia care le-a blocat sau pot fi toate deblocate in cazul unui COMMIT sau unui ROLLBACK, in functie de modelul de blocare folosit.

6.2. Gestiunea tranzactiilor

In paragraful anterior tranzactia era definita ca o executie a unui program. In fapt un program care interactioneaza cu o baza de date contine de obicei nu o singura tranzactie ci o succesiune de tranzactii care nu se intersecteaza, fiecare dintre ele fiind finalizata fie prin comiterea modificarilor efectuate (ele devin definitive in baza de date) fie prin revocarea lor (modificarile sunt anulate). Dupa terminarea unei tranzactii celelalte operatii asupra bazei de date apartin tranzactiilor urmatoare.

Cum singurele operatii care sunt importante din punct de vedere al interactiunii dintre program si sistemul de gestiune sunt cele de citire/scriere si cele conexe (blocare, deblocare, comitere si revocare) putem defini o tranzactie si ca o succesiune de operatii de acest tip.