Unele anomalii în execuția tranzacțiilor. Protocolul de închidere în două faze.

Unele probleme de terminologie

În mod convențional, se acceptă că o tanzacție **„completată”**, sau **„completă”**, este unitatea de prelucrare executată integral, ea reprezentând parcurgerea şi prelucrarea corectă a tuturor componentelor programului pe care îl reprezintă. În mod normal, o tranzacție completată lasă baza de date asupra căreia a acționat într-o stare de coerență. Reamintim aici faptul că SGBD trebuie să asigure patru proprietăți ale tranzacțiilor: atomicitate, consistență (sau coerență), izolare şi durabilitate (ACID).
Tranzacția **„abandonată”** este o tranzacție **incompletă**, sau **necompletată**, respectiv executată fără succes. Dintre cauzele abandonării unei tranzacții, menționăm:
a). ca urmare a întâlnirii unei anomalii în timpul execuției, SGBD termină tranzacția fără succes. Atunci când sursa anomaliei a fost internă, iar întreruperea execuției (adică abandonarea) a fost decisă de SGBD, execuția tranzacției este reluată automat.
b). În timpul execuției unei tranzacții sau a mai multor tranzacții, are loc o defecțiune de genul întreruperii sursei de alimentare cu energie.
c). Tranzacția în curs de execuție decide „singură” întreruperea prelucrării, ca urmare a întâlnirii unei situații neprevăzute, cum ar fi, de exemplu, citirea unei valori neaşteptate. În cazul abandonării execuției, toate rezultatele acțiunilor efectuate şi terminate până în acel moment sunt anulate („distruse” – „undone”), Într-o asemnea situație, tranzacția respectivă trebuie să specifice acțiunea sa finală ca „abandonare” sau „reîntoarcere” („rollback”).
„Comiterea”, sau încheierea cu succes , este situația în care o tranzacție este „completată” cu succes. Ca acțiune finală a sa, tranzacția trebuie să specifice starea „Commit”.
În contextul de mai sus, o planificare este o listă de acțiuni (citiri, înregistrări, abandonări, comiteri) care apatțin unei mulțimi de tranzacții.

Anomalii legate de execuția intercalată a tranzacțiilor. Citirea datelor ne-comise.

Ne vom referi în continuare la citirea unor date „ne-comise”, fapt ce poate fi considerat o situație de **conflict înregistrare-citire**. O tranzacție T2 poate citi un obiect A al unei baze de date modificat anterior de tranzacția T1, care însă nu fusese comisă, deci care nu specificase prin mesajul „commit” faptul că ajunsese în faza de completare. Această situație se asociază uneori cu termenul **„citire impură”**, efectuată de către tranzacție T2. Pentru ilustrare, fie exemplul care urmează:

T1	T2
Read A	
Write A	
	Read A
	Write A
	Read B
	Write B
	Commit
Read B	
Write B	
Commit	

Fig. 1

Pentru a aprofunda maniera de execuție a aceste planificări, să admitem că A şi B sunt două conturi de bancă, având conținutul A= 200, B= 100. Presupunem, de asemenea, că această stare a bazei de date este coerentă. Acceptăm, în continuare că în ordinea T1, T2 tranzacția T1 extrage din A 100 de unități (deci debitează A), pe care le introduce în B (aşadar, creditează B cu 100 unități). Apoi, la rândul său, tranzacția T2 incrementează valorile elementelor cu 6 %, care reprezintă o dobândă. Mai jos, este reprezentată o execuție de planificare în care sunt specificate operațiunile descrise.

T1	A= 200 B =100	T2
Read A		
A= A- 100		
Write A (A devine 100)		
		Read A
		A= A+0,06A
		Write A (A devine 106)
		Read B
		B= B+ 0,06B
		Write B (B devine 106)
		Commit
Read B		
B= B+ 100		
Write B (B devine 206)		
Commit		

Fig. 2

Rezultatul obținut prin planificarea de sus este diferit de cel la care ne pot conduce ambele execuții seriale posibile, adică T1, T2, sau T2, T1. În situația de aici se ajunge dacă o tranzacție foloseşte informațiile date de altă tranzacție, înainte ca ultima să fi fost „comisă". Observăm că în cazul tratat, tranzacția T2 foloseşte valoarea calculată pentru A de tranzacția T1, înainte de comiterea acesteia.
Planificarea anterioară ne dă valorile A= 106 şi B= 206 pentru cele două conturi. Dacă însă ar fi fost folosită planificarea serială T1, T2, valorile obținută pentru cele două obiecte de date ar fi fost A= 106 şi B= 212. În cazul planificării seriale T2, T1, valorile finale ar fi fost A= 112 şi B= 206.

Citire nerepetabilă

Citirea nerepetabilă constitue o altă categorie de conflict citire- înregistrare. Acest tip de anomalie apare tot pe fondul suprapunerii, sau intercalării, încercărilor unor tranzacții concurente de a iniția operațiuni conflictuale. În cazul examinat, o tranzacție T2 poate schimba valoarea obiectului de date A care a fost anterior citit de o altă tranzacție T1, în timp ce această ultimă tranzacție este încă în execuție, cu alte cuvinte nu este nici comisă, nici abandonată.
Mai întâi, va fi prezentat cazul simplu, convențional, al execuției pur seriale a două tranzacții T1 şi T2. Prima tranzacție citeşte valoarea unui obiect de date A, iar a doua tranzacție citeşte A, modifică valoarea acestuia şi înscrie în memoria centrală noua valoare:

T1	T2	În decursul execuției planificării
.		reprezentate aici, tranzacția T1 nu
.		mai acționează asupra obiectului A
Read A		după ce a efectuat prima citire, deci
.		nu va citi din nou, nu va înregistra
.		sau efectua modificări. Dacă inițial

.	valoarea obiectului A a fost 10, la
.	sfârşitul execuţiei A= 15.
Read A	
A= A+ 5	
Write A	

Fig. 3

.

Să presupunem acum că tranzacţiile T1 şi T2 sunt intercalate în decursul execuţiei, iar tranzacţia T1 va citi obiectul A pentru a doua oară, după ce valoarea acestuia a fost citită, modificată şi înregistrată în memoria centrală de către tranzacţia T2:

T1	T2	Pe parcursul execuţiei
.		alăturate, la a doua citire pe
.		care o efectuează, tranzacţia
Read A (aici A= 10)		T1 găseşte un rezultat diferit
.	.	faţă de cel aflat la prima citire
.	.	pe care o realizase, chiar dacă
	Read A (şi aici A= 10)	această tranzacţie nu modifi-
	A= A+5 (A devine 15)	case obiectul A între timp. În
	Write A (A=15)	literature de specialitate, acest
	.	gen de operaţie este denumit
	.	"citire nerepetabilă".
.		
.Read A (aici A= 15)		

Fig. 4

Protocolul de închidere "strictă" în două faze (2PL "strict")

Potrivit unei definiţii acceptate, protocolul de închidere este constituit de un grup de reguli care, ca urmare a acţiunii SGBD, trebuie respectate de fiecare tranzacţie a unei planificări, pentru a asigura obţinerea unui efect identic cu cel al execuţiei tuturor tranzacţiilor într-o ordine serială oarecare, chiar şi prin intercalarea acţiunilor unora din tranzacţii. Regulile protocolului sunt enunţate în continuare.

Regula 1. Pentru a citi (respectiv, pentru a modifica) valoarea unui obiect de date, o tranzacţie trebuie mai întâi să solicite o închidere partajată (respectiv o închidere exclusivă) pe acel obiect.
Observăm că o tranzacţie care deţine dreptul de închidere exclusivă aspra unui obiect de date, poate de asemenea citi acel obiect, fără a fi necesară solicitarea unei închideri partajate adiţionale. Aşadar, în reprezentarea din Fig. 5, care urmează:

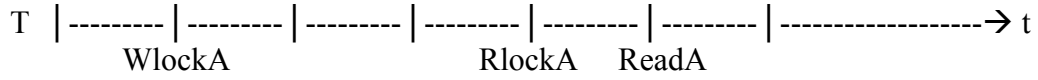

Fig. 5

tranzacţia T poate executa operaţia Read A fără a mai solicita închiderea partajată RlockA.
Totodată, mai remarcăm că din momentul solicitării unui drept de închidere, tranzacţia respectivă este "suspendată" până când SGBD îi va atribui dreptul de închidere. Desigur, SGBD păstrează evidenţa închiderilor acordate şi nu permite niciunei tranzacţii să deţină (sau să obţină) închiderea

partajată sau exclusivă asupra unui obiect de date, dacă o tranzacție dată posedă dreptul de închidere exclusivă pe acel obiect.

Regula 2. Toate închiderile deținute de către o tranzacție sunt eliberate atunci când tranzacția este comisă (completată).
Trebuie notat faptul că utilizatorul bazei de date nu trebuie să se îngrijească de acordarea drepturilor de închidere sau deschidere, această operațiune fiind realizată automat de către SGBD.

Protocolul 2PL strict asigură serializabilitatea planificărilor care sunt executate cu respectarea regulilor menționate şi, totodată, nu permite apariția unor anomalii de genul celor descries anterior. Ca urmare, acest protocol reprezintă cea mai frcevent utilizată metodă de închidere a accesului la tranzacții.
Protocolul de închidere asigură numai intercalarea sigură a tranzacțiilor. Dacă două tranzacții au acces la acelaşi obiect de date, iar una din tranzacții încearcă modificarea valorii obiectului, acțiunile celor două tranzacții vor fi ordonate serial de către protocol: mai întâi vor fi efectuate (până la completare, sau comitere) acțiunile tranzacției care a obținut prima dreptul de închidere pe acest obiect, iar după eliberarea închiderii poate începe lucrul cea de a doua tranzacție.
Evident, dacă cele două tranzacții au acces la două granule complet independente ale bazei de date , ele pot obține în manieraă concurentă dreptul de închidere pe care îl doresc şi pot începe execuția fără alte probleme. Deci tranzacțile inchid independent, simultan, cele două obiecte.

Alte notații obişnuite în cazul acestui protocol: O- obiect de date (sau, cu un alt termen, articol), S- cerecre de închidere, S(O)- scriere echivaentă cu Rlock O. Pentru a arăta că dreptul de închidere partajată pe obiectul O este acordat tanzacției T, se foloseşte notația $S_T(O)$. Pentru cererea de închidere exclusivă – echivalentă cu Wlock O- se poate folosi notația $X_T(O)$.

Să reluăm execuția din Fig. 1:

T1	A=B	T2
Read A		
Write A		
		Read A
		Write A
		Read B
		Write B
		Commit
Read B		
Write B		
Commit		

În ipoteza că pentru execuție este folosit protocolul 2PL strict, intercalarea de acțiuni de mai sus nu este permisă. Execuția care ilustrază prelucrarea cu acest protocol va fi:

T1	T2		T1	T2
X(A)			Wlock A	
R(A)		sau, cu altă notație:	Rlock A	
W(A)			Write A	

Fig. 6

După consumarea acțiunilor din execuția de mai sus, tranzacția T2 va solicita acordarea dreptului de închidere pe obiectul A. Această cerere nu va fi însă satisfăcută decât după ce tranzacția T1 eliberează închiderea exclusivă asupra obiectului A, aşadar SGBD suspendă tranzacția T2. În continuare, T1 solicită dreptul de închidere exclusivă asupra obiectului de date B, îl obține, citeşte B şi eventual îl modifică, după care înregistrează noua valoare pentru B, apoi comite şi în acest moment este eliberată închiderea care îi aparține. Acum, SGBD poate acorda dreptul de închidere exclusivă tranzacției T2, care intră în lucru. În acest exemplu, constatăm că acțiunea protocolului 2PL strict conduce la o execuție serială a tranzacțiilor, reprezentată în Fig. 7:

T1	T2
X(A)	
R(A)	
W(A)	
X(B)	
R(B)	
W(B)	
Commit	
	X(A)
	R(A)
	W(A)
	X(B)
	R(B)
	W(B)
	Commit

Fig. 7

În acelaşi timp, protocolul 2PL strict permite o intercalare a tranzacțiilor T1 şi T2, ca mai jos:

T1	T2
S(A)	
R(A)	
	S(A)
	R(A)
	X(B)
	R(B)
	W(B)
	Commit
X(C)	
R(C)	
W(C)	
Commit	

Fig. 8

sau, cu altă notație:

T1	T2
Rlock A	
Read A	
	Rlock A
	Read A
	Wlock B
	Read B
	Write B
	Commit
Wlock C	
Read C	
Write C	
Commit	

Fig. 9

Se observă că acțiunile tranzacției T1, efectuate după comiterea tranzacției T2, au loc asupra unui alt obiect de date, C, care este complet independent față de A şi B. Desigur, maniera de execuție a

planificării anterioare poate fi analizată mai profund, încercând să stabilim ce evoluție au diferitele obiecte de date după parcurgerea operațiilor în care sunt prelucrate.

Protocolul de închidere „ne-strictă" în două faze (2PL „ne-strict")

Protocolul de închidere „ne-strictă" în două faze este o variantă a protocolului 2PL strict, bazându-se pe reguli mai puțin severe decât acesta. Ne reamintim, în acest context, că a doua regulă a protocolului 2PL strict impunea ca o tranzacție să renunțe la toate drepturile de închidere obținute numai după completarea sa, deci după încheierea tranzacției, cu toate acțiunile sale, prin una din comenzile „commit" sau „abort". La această regulă, exprimată în forma de mai sus, se renunță în protocolul 2PL ne-strict, apelându-se la o regulă nouă, formulată în cele ce urmează. Pentru o prezentare mai completă, vom relua şi prima regulă a protocolului 2PL strict, care este valabilă şi pentru varianta 2PL ne-.strict. Formularea regulilor protocolului 2PL ne-strict este următoarea:

Regula 1 (aceeaşi ca în 2PL strict) : dacă o tranzacție urmează să citească (sau, respectiv, să modifice) un obiect de date, va solicita mai întâi dreptul de închidere partajată (pentru care folosim notația Rlock sau notația S) pe obiectul avut în vedere, respectiv dreptul de închidere exclusivă (cu notația Wlock sau X).

Regula 2 (specifică protocolului 2PL ne-strict): odată ce a eliberat orice fel de închidere, o tranzacție nu mai poate solicita alte drepturi de închidere, adiționale.

Se poate demonstra că, similar protocolului cu închidere strictă, folosirea protocolului 2PL ne-strict conduce numai la planificări serializabile, pentru care graful de precedență nu are contururi închise. În adevăr, se poate înțelege intuitiv că o ordonare serială echivalentă execuției unui grup de tranzacții (fie, de exemplu, T1 şi T2), este obținută urmând ordinea în care tranzacțiile intră în faza în care eliberează închiderile:dacă T2 citeşte sau înregistrează un obiect înregistrat de T1, atunci tranzacția T1 a eliberat dreptul de închidere ce îl avea asupra obiectului considerat, înainte ca tranzacția T2 să fi solicitat drept de închidere pe acest obiect. În acest mod, tranzacția T1 precede tranzacției T2. Într-o manieră asemănătoare, se poate constata că dacă tranzacția T2 înregistrează informație într-un obiect care a fost citit anterior de T1, atunci suntem de asemenea în situația în care tranzacția T1 precede tranzacția T2.
Având în vedere proprietățile celor două protocoale de închidere prezentate, este util să dormulăm câteva observații privind natura unor execuții de planificări. Dacă în interiorul unei execuții date, o valoare înregistrată de tranzacția T nu este citită sau supra-înregistrată de către o altă tranzacție înainte de comiterea sau abandonarea tranzacției T, vom asocia cu execuția de planificare respectivă adjectivul „strictă"
Execuțiile stricte de planificare pot fi refăcute (sunt recuperabile), ele nu solicită abandonări „în cascadă", iar rezultatele acțiunilor tranzacțiilor abandonate pot fi eliminate prin restaurarea valorilor originale ale obiectelor de date modificate. Pentru a ilustra ultimele considerațiuni, să descriem unele consecințe ale eliminării rezultatelor acțiunilor unor tranzacții. Fie cazul în care o tranzacție T2 supra-înregistrează valoarea unui obiect A care a fost anterior modificat de tranzacția T1, supra-înregistrarea având loc în timp ce tranzacția T1 este încă în lucru, iar apoi T1 abandonează execuția. Toate schimbările efectuate de tranzacția T1 asupra obiectelor bazei de date sunt anulate prin refacerea (restaurarea) valorii oricărui obiect pe care tranzacția l-a modificat, aducându-l la valoarea avută înainte ca tranzacția T1 să efectueze schimbările menționate. Atunci când T1 abandonează execuția, iar modificările efectuate au fost anulate în modul descris, modificările făcute de tranzacția T2 vor fi de asemenea pierdute, chiar dacă această ultimă tranzacție comite. Vom concretiza succint raționamentul prezentat. Dacă un obiect de date A are inițial valoarea 10, iar apoi este modificat de tranzacția T1 la o nouă valoare 12 şi de către tranzacția T2 la valoarea 15, dacă T1 abandonează obiectul revine la valoarea 10. În acest lanț de operațiuni, chiar dacă tranzacția T2

comite, modificarea pe care ea a făcut-o în valoarea obiectului A este pierdută. Acest comportament poate fie evitat dacă planificrea se desfăşoară în coordonarea protocolului 2PL strict.
Protocolul 2PL strict este superior variantei ne-stricte deoarece garantează faptul că fiecare execuție permisă este strictă, fiind, de asmenea, serializabilă „la conflict”. Această proprietate a protocolului 2PL strict decurge din faptul că atunci când o tranzacție înregistrează un obiect de date în coordonarea acestui protocol, ea va menține dreptul de închidere exclusivă până la comitere sau abandonare.. Prin urmare, nicio altă tranzacție nu va primi drept de citire sau de înregistrare asupra obiectului, atât timp cât tranzacția nu este completată.