Tipuri de concurenţă vis-a-vis de modificarea datelor unei aplicaţii (optimistă, pesimistă, ultimul câştigă)

Datorită faptului că datele dintr-o aplicaţie pot fi vizualizate şi modificate de către mai mulţi utilizatori în acelaşi timp pot apărea diferite probleme, în special la modificarea datelor. Pentru a rezolva aceste probleme s-au găsit mai multe strategii de concurenţă, iar în continuare vom discuta despre 3 dintre ele.

1. Optimistă: atunci când se face un update pe aplicaţie, se verifică dacă au fost modificate acele date care urmează să fie schimbate de când utilizatorul le-a preluat. Dacă se fac 2 modificări asupra aceleaşi date atunci una nu trebuie să o suprascrie pe cealaltă, iar utilizatorii trebuie să afle dacă update-ul a avut loc cu succes. În acest fel toate datele sunt disponibile tuturor utilizatorilor, dar este mai dificil să se lucreze în acest fel dacă se fac multe modificări în interval de timp scurt. Există mai multe variaţiuni pe această temă, pe lângă cea de mai sus.

   Se pot compara toate câmpurile de pe rândul pe care se efectuează modificări, deoarece dacă User 1 face update la câmpul Nume, iar User 2 la câmpul prenume nu va apărea conflict, iar datele vor fi eronate. Este o variantă mai costisitoare, dar mai bună decât cea de sus.

   Cea mai bună variantă, din punctul meu de vedere, este: se mai adaugă un câmp la rând pentru Timestamp care se modifică la fiecare update pe oricare dintre coloanele rândului respectiv. Asta înseamnă că la atunci când User 1 face o modificare pe rândul respectiv şi User 2 încearcă imediat după să facă modificarea va primi o eroare de la server deoarece valoarea câmpului Timestamp s-a schimbat de când a fost începută prelucrarea sa de către User 2 şi funcţionează cu oricâte câmpuri modificate.

   Este mai simplu de implementat şi mult mai scalabilă decât varianta pesimistă, sunt puţine riscuri de deadlock, dar de asemenea e ceva mai puţin sigură.

2. Pesimistă: când o aplicaţie/utilizator doreşte să modifice o dată va bloca accesul altora la acea dată până termină de modificat. Este uşor de implementat deoarece serverele susţin mecanisme de locking. Pentru a face modificări pe baza de date utilizatorul trebuie să blocheze accesul celorlalţi, iar serverul va informa ceilalţi useri dacă vor încerca să acceseze aceleaşi date, iar nimic nu va putea ignora lock-ul. Dar, nu este o metodă scalabilă deoarece necesită multă resurse, toţi utilizatorii trebuie să aibă o conexiune cu baza de date deschisă.

De asemenea, uneori datele sunt blocate pentru mai mult timp din diferite motive şi chiar o simplă apăsare de buton de care s-a uitat poate bloca activitatea. Există şi o variantă de softlock care marchează locurile unde se fac modificări (de exemplu Google Sheets) sau se blochează accesul din setările paginii.

3. Ultimul câştigă: nu este o strategie în adevăratul sens al cuvântului, ci mai degrabă un compromis, ultimele date trecute sunt cele salvate. Această tehnică este folosită în locuri unde informaţia este împărţită în mai multe locuri şi nu sunt mai mulţi utilizatori ce încearcă să facă update în acelaşi timp, pe aceeaşi dată. De folos mai ales în live-stream-uri scurte.

Ce probleme mari putem întâmpina când blocăm accesul cu concurenţa pesimistă:

Deadlock - apare atunci când 2 sau mai multe procese aşteaptă unul după celălalt să deblocheze o resursă, sau mai multe procese aşteaptă deblocarea în lanţ circular a unei resurse. Această problemă este frecventă în multiprocesare unde unele resurse sunt împărţite. Unele computere au implementat un sistem de hard lock care garantează accesul exclusiv la o resursă.

Livelock - se referă la procese şi la faptul că starea lor se poate schimba constant, însă ele nu progresează, ci rămân într-o stare de indecizie. În acest fel aplicaţia se blochează dacă niciun proces nu devine prioritar şi îşi termină execuţia. Un exemplu din natură îl poate reprezenta o intersecţie nesemaforizată în care ajung maşini din fiecare direcţie şi nu se ştie cine are exact prioritate şi toţi aşteaptă.

Când vine vorba de aplicaţii care lucrează cu baze de date simple metodele standard de rezolvare a concurenţei funcţionează foarte bine deoarece datele pe care utilizatorul le vede reprezintă un singur rând în baza de date.

Dar, de obicei se lucrează cu aplicaţii mai complexe, iar aici intervine partea mai dificilă. Luând un exemplu din lumea reală : pentru vânzarea unui produs sunt implicate mai multe departamente şi varianta cu un produs = 1 rând într-o tabelă nu se poate aplica. Dacă pe un singur departament se pot găsi strategii care să evite acest lucru, când sunt implicate mai multe departamente nu mai funcţionează aşa.

Un posibil mod de lucru pentru exemplul de mai sus :

1. Se începe conexiunea/tranzacţia cu baza de date
2. Se verifică versiunea rândului din tabela de comenzi produse
3. Update pe header-ul rândului, trebuie făcut chiar dacă nu se efectuează vreo schimbare la rând, pentru a updata versiunea rândului
4. Update, delete sau insert pe toate rândurile din tabela pentru detalii comandă

5. Commit/finalizarea tranzacţiei