Popovici Daniela
Anul 4, IS

# Tipuri de concurență vis-a-vis de modificarea datelor unei aplicații( optimistă, pesimistă, ultimul câștigă)

~tema 2 DATC~

Într-un sistem sau o aplicație cu mai mulți utilizatori, concurența este una dintre cele mai importante probleme care trebuie rezolvate. În contextul unei aplicații software, concurența se referă la gestionarea faptului că mai mulți utilizatori încearcă să acceseze aceleași date, în același timp.

De exemplu, se consideră un sistem de procesare a comenzilor care are mai mulți utilizatori. Sistemul le permite utilizatorilor să adauge sau să editeze comenzi. Adăugarea comenzilor noi nu prezintă o problemă deoarece fiecare comandă generează o nouă înregistrare. Astfel mai mulți utilizatori pot insera înregistrări noi fără ca acestea să interfereze. În schimb, editarea comenzilor poate conduce la probleme de concurență. Când un utilizator deschide o comandă pentru a o edita, acestuia ii se deschide o fereastră de dialog care conține informații despre comandă. După ce acesta face modificări, acestea sunt trimise spre server și înregistrarea este updatată. Problema apare dacă doi sau mai mulți utilizatori incearcă să modifice aceași comandă. Pentru a rezolva această problemă, aplicația are un anumit design care prevede cum se gestionează concurența.

Există 3 strategii principale care se ocupă de problema concurenței:

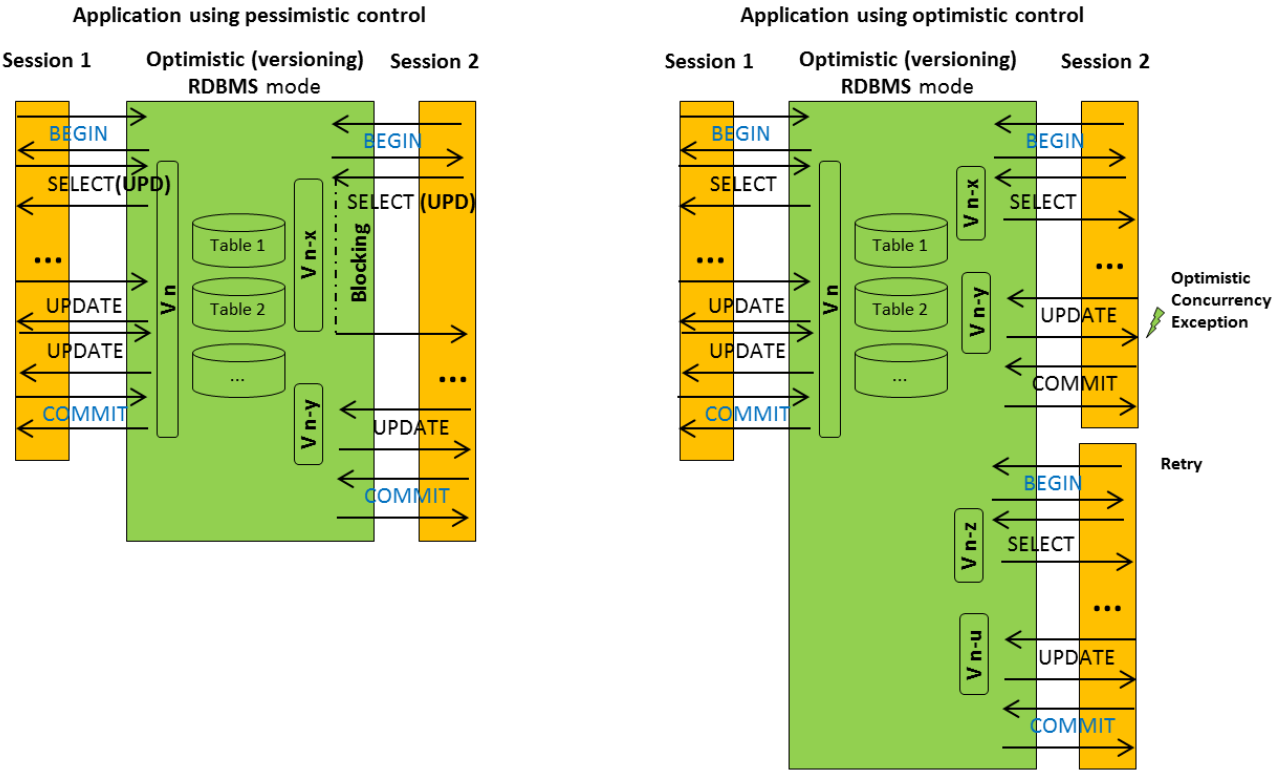
1. *Concurență optimistă*

- În modelul de concurență optimistă, utilizatorii au întotdeauna permisiunea de a citi datele și de a le modifica/actualiza
- Această abordare a problemei permite mai multor utilizatori să vizualizeze date, față de cât permite modelul pesimist
- În cazul în care utilizatorul încearcă să salveze datele, sistemul verifică dacă datele au fost actualizate de oricine altcineva de la data ultimei citiri a datelor; dacă au fost făcute modificări atunci actualizarea eșuează
- Este incomod atunci când un utilizator își petrece timpul pentru a actualiza o înregistrare și apoi află că modificarea lui nu poate fi salvată; înregistrările trebuind recuperate din nou și modificările efectuate din nou

- De exemplu, dacă doi utilizatori care vizualizează o pagină Wiki fac o actualizare a aceleiași pagini, atunci platforma Wiki trebuie să se asigure că cea de a doua actualizare nu suprascrie prima actualizare și că ambii utilizatori înțeleg dacă actualizarea lor a avut succes sau nu
- De obicei, acest model se folosește când este mai puțin probabil ca mai mulți utilizatori să încerce să editeze aceleași date în același timp
- Această strategie este utilizată cel mai adesea în aplicațiile Web

2. *Concurență pesimistă*

- Acest model de concurență plasează blocări pe date
- Dacă un utilizator are o înregistrare deschisă și alți utilizatori încearcă să citească acele date atunci sistemul respinge cererea de citire a datelor
- În momentul în care un utilizator deschide o comandă pentru a o edita acesta obține blocarea înregistrării; utilizatorii ulteriori care încearcă să deschidă comanda vor primi un mesaj care să îi anunțe că comanda este editată în prezent de un alt utilizator și că vor trebui să aștepte până când primul utilizator își salvează modificările sau anulează operația de editare
- Acest model are și dezavantaje și anume faptul că sistemul este mai puțin convenabil de utilizat deoarece majoritatea utilizatorilor sunt împiedicați să vizualizeze datele pe care le-a deschis un singur utilizator; în plus, implementarea devine mai greoaie din cauză că sistemul trebuie să gestioneze blocările de înregistrări
- Modelul de concurență pesimistă este cel mai bun în situațiile în care este foarte probabil ca mai mulți utilizatori să încerce să editeze aceleași date în același timp



3. *Ultimul câştigă*

- Nu se face vreo verificare
- Această abordare permite efectuarea oricăror operații de actualizare fără să verifice dacă datele au fost actualizate de la prima citire a lor de către aplicație
- De obicei, este utilizată în cazul în care datele sunt împărțite astfel încât să nu existe nici o probabilitate ca mai mulți utilizatori să aibă acces la aceleași date
- Poate fi utilă această abordare în cazul procesării fluxurilor de date cu durată scurtă de viață