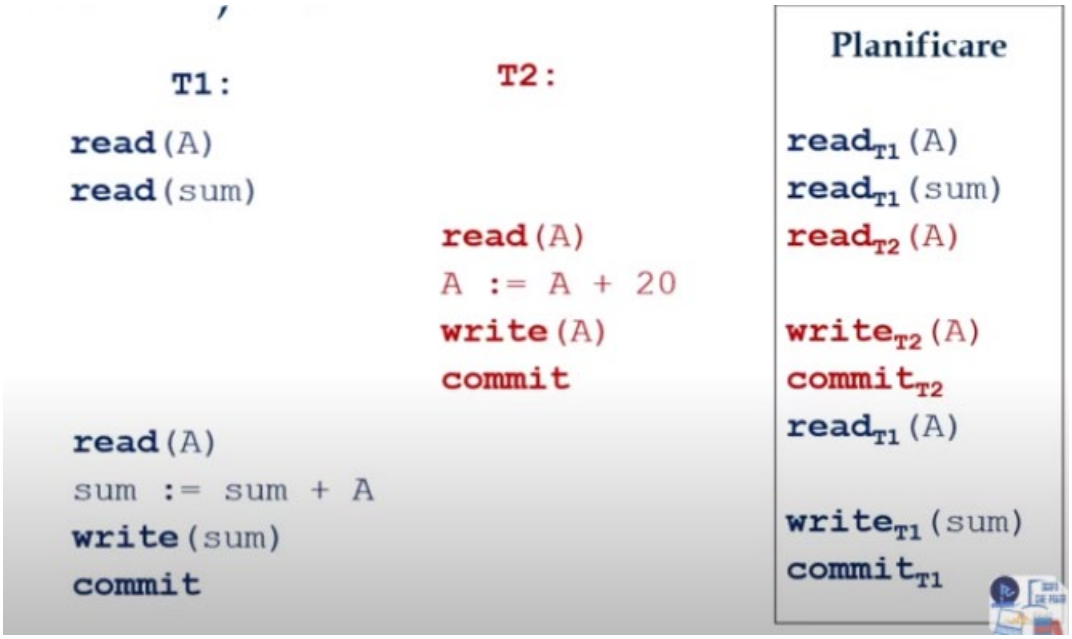
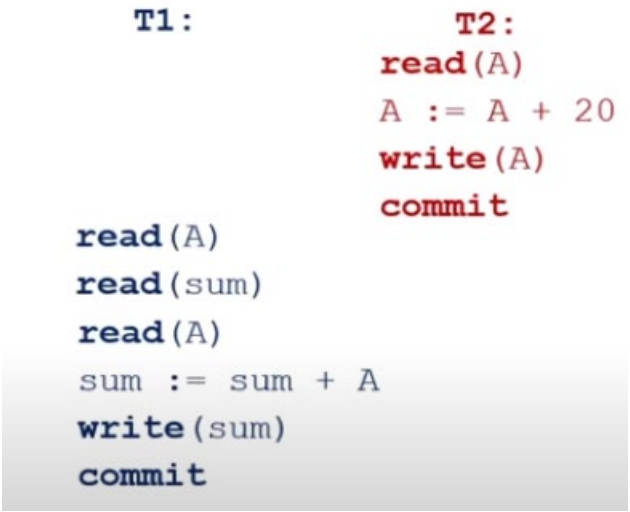
Planificarea tranzacțiilor

SGBD ignoră operații care nu sunt de tip Read / Write, deoarece acestea nu provoacă o alterare a datelor.

O *planificare a tranzacțiilor* se referă la modul/**ordinea** în care sunt intercalate operațiile Read / Write / Abort / Commit ale unei tranzacții cu aceleași operații ale altei tranzacții ce se execută în același timp.

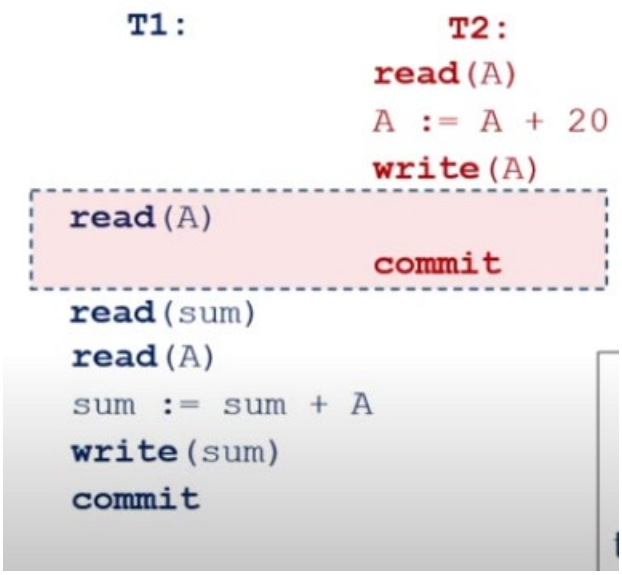


Planificare serială – orice planificare ce nu intercalează acțiuni ale mai multor tranzacții



Planificare non-serială

- acțiunile mai multor tranzacții concurente se întrepătrund
- o condiție ca să avem o astfel de planificare este ca ultima operație a fiecăreia dintre tranzacții să se execute după prima operație a celeilalte tranzacții



Rezultatul execuției unei planificări trebuie să lase baza de date într-o stare consistentă.

Planificări echivalente = efectul a două planificări asupra bazei de date este identic

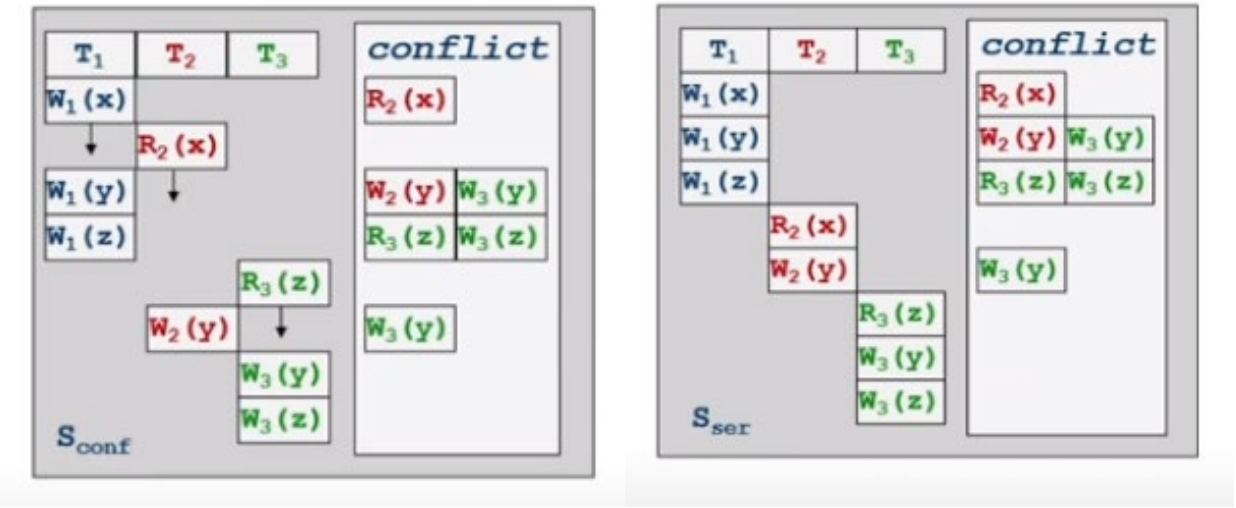
Planificare serializabilă = efectul execuției concurente a mai multor tranzacții este echivalent cu execuția serială a tranzacțiilor.

!!! Nu ar trebui să executăm planificări care nu sunt serializabile.

Operație conflictuală = orice pereche de operații în care cel puțin una este operație de **write.**

Care este regula pe care trebuie să o urmăm astfel încât ca planificarea noastră non-serială să fie serializabilă?

Ordinea în care se execută operațiile conflictuale să fie aceeași, și într-o planificare, și în cealaltă.



Două planificări sunt *conflict-echivalente* dacă:

- implică aceleași tranzacții
- fiecare pereche de acțiuni conflictuale este ordonată în același mod

O planificare este *conflict-serializabilă* dacă e conflict-echivalentă cu o planificare serială.

Serializabilitate:

- obiectivul serializabilității este găsirea unei planificări non-seriale care permite execuția confurentă a tranzacțiilor fără ca acestea să interfereze și, astfel, să conducă la o stare a unei baze de date la care se poate ajunge și printr-o execuție serială
- previne apariția inconsistențelor generate de interferența tranzacțiilor

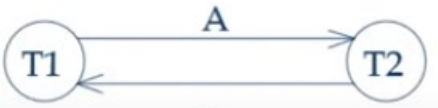
Conflict-serializabilitate:

- un mod de a observa dacă o planificare este conflict-serializabilă sau nu e prin trasarea unui graf de precedență:
  - graf orientat
  - fiecare nod din graf reprezintă o tranzacție
  - ducem un arc de la T_i la T_j dacă avem o operație în T_j care e conflictuală cu una din T_i și care urmează după operație din T_i
- dacă avem circuite în graf, atunci planificarea nu este conflict-serializabilă

■ Planificare ce nu este conflict-serializabilă:

T1: R(A), W(A), R(B), W(B)

T2: R(A), W(A), R(B), W(B)



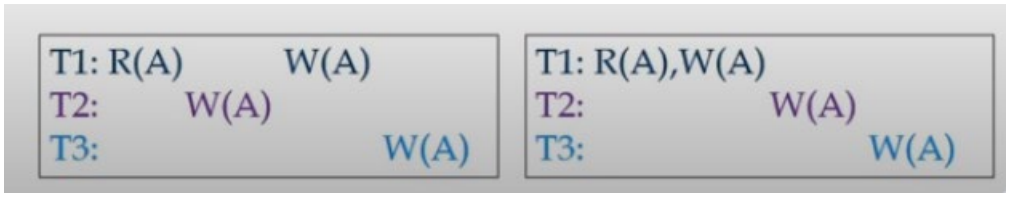
Graf de precedență

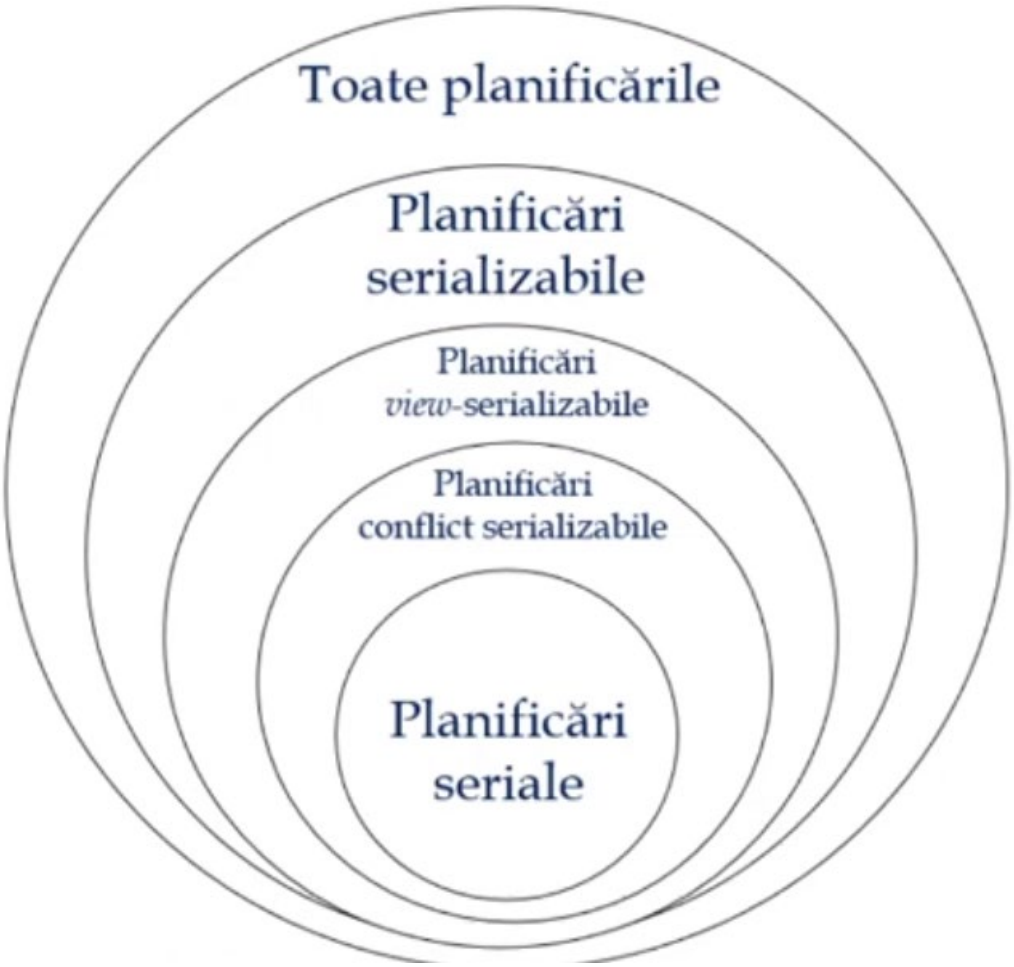
Algoritmul de Testare a Conflict-Serializabilității lui S:

1. Pentru fiecare tranzacție T_i din **S** de crează un nod etichetat T_i în graful de precedență.
2. Pentru fiecare **S** unde T_j execută un Read(x) după un Write(x) executat de T_i crează un arc (T_i, T_j) în graful de precedență
3. Pentru fiecare caz în **S** unde T_j execută un Write(x) după un Read(x) executat de T_i crează un arc (T_i, T_j) în graful de precedență
4. Pentru fiecare caz în **S** unde T_j execută un Write(x) după un Write(x) executat de T_i crează un arc (T_i, T_j) în graful de precedență

View-serializabilitatea:

- planificările S_1 și S_2 sunt view-echivalente:
  - dacă T_i citește valoarea inițială a lui A în S_1, atunci T_i de asememea citește valoarea inițială a lui A în S_2
  - dacă T_i citește valoarea lui A modificată de T_j în S_1, atunci T_i de asemenea citește valoarea lui A modificată De T_j în S_2
  - dacă T_i modifică valoarea finală a lui A în S_1, atunci T_i de asemenea modifică valoarea finală a lui A în S_2




Toate planificările
Planificări serializabile
Planificări *view*-serializabile
Planificări conflict serializabile
Planificări seriale