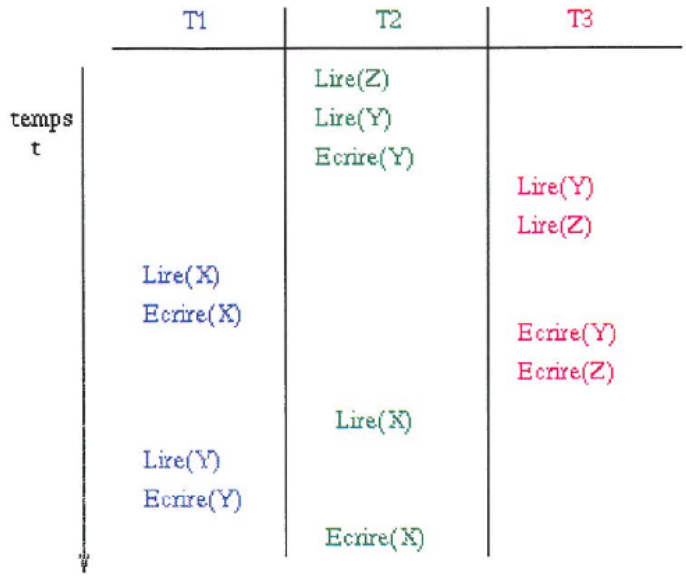
REALISATION D'UN GRAPHE DE SERIALISABILITE

Comme vu precedemment, le graphe de serialisabilite traduit les relations de precedence entre les transactions impliquees dans une execution. Ces relations dependent fortement de l'ordre d'execution des operations elementaires des transactions.

Nous allons voir sur trois exemples, comment se traduit une execution en termes de relations de precedence.

Exemple 1.

Schema 1: Transactions en serie



Dans un premier temps, nous etablissons pour chaque objet la liste des transactions qui accedent a l'objet par ordre chronologique.Dans un second temps, nous appliquons l'algorithme de serialisabilite.

Pour L'objet X:

Lecture de T1, Ecriture de T1, Lecture de T2, Ecriture de T2.

Relation de precedence: T1 precede T2.

Pour l'objet Y:

Lecture de T2, Ecriture de T2, Lecture de T3, Ecriture de T3, Lecture de T1, Ecriture de T1.

Relation de precedence: T2 ecrit avant T3 donc T2 precede T3;

T2 ecrit avant T1 donc T2 precede T1;

T3 ecrit avant T1 donc T3 precede T1.

Pour l'objet Z:

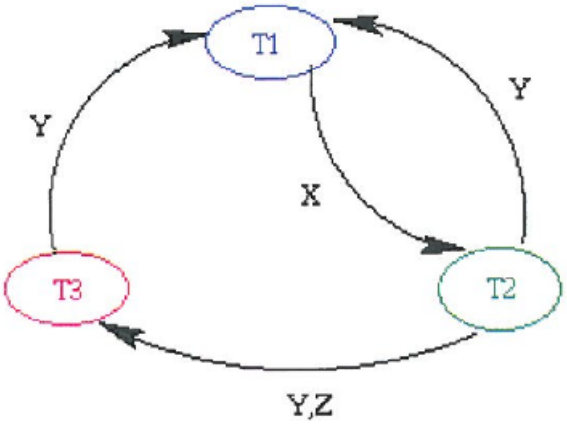
Lecture de T2, Lecture de T3, Ecriture de T3.

Relation de precedence: Le fait que T2 lise avant que T3 ne lise ne provoque pas de conflit: les deux premieres operations de la liste precedente ne signifie pas que T2 precede T3. Cependant, la precedence doit etre exprimee ici, parce que T3 fait par la suite une operation d'ecriture sur Z.

CONSTRUCTION DU GRAPHE DE PRECEDENCE

1. La construction en est simple, une fois les relations de precedence etudiees.
2. Toutes les transactions sont representees par des noeuds.
3. On indique que T1 precede T2 en dessinant une fleche de T1 vers T2.
4. Pour plus de clarte, on peut mettre sous chaque fleche, l'objet a partir duquel a ete deduite la relation representee par la fleche.

Le graphe de precedence ressempble alors a ce qui suit:



X(T1 ⟶ T2); Y (T2 ⟶ T1)

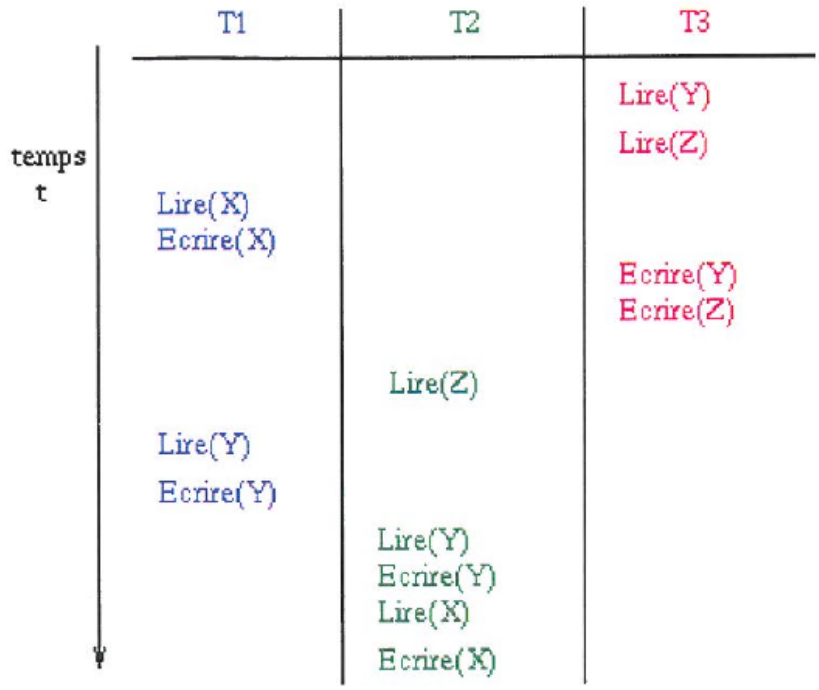
X(T1 ⟶ T2); YZ (T2 ⟶ T3); Y (T1 ⟶ T3);

Conclusion: On ne peut pas trouver d'execution de T1, T2, T3 en serie, car il y a presence d'un cycle (deux cycles) dans le graphe: voir schema precedent.

Ceci signifie que si l'on effectue dans cette ordre, en multitache, les transactions T1, T2, T3, on n'obtiendra pas sauf cas particulier, le resultat voulue . Ce resultat devrait etre le meme que le resultat detrois transformations finies, atomiques de la base de donnees executeesdans un ordre quelconque: Qu'on fasse T1,T2,T3 ou T2, T3, T1, on obtiendra le meme etat final de la base de donnees.

Exemple 2.

Nous prenons les memes transactions mais executees en parallele dans un ordre tout a fait different:



Comme precedemment, dans un premier temps, nous etablissons pour chaque objet la liste des transactions qui accedent a l'objet par ordre chronologique. Et dans un second temps, nous appliquerons l'algorithme de serialisabilite.

Pour L'objet X:

Lecture de T1, Ecriture de T1, Lecture de T2, Ecriture de T2.

Relations de precedence: On determine sans difficulte que T1 precede T2.

Pour l'objet Y:

Lecture de T3, Ecriture de T3, Lecture de T1, Ecriture de T1, Lecture de T2, Ecriture de T2.

Relations de precedence: T3 ecrit avant T1 donc T3 precede T1;

T3 ecrit avant T2 donc T3 precede T2;

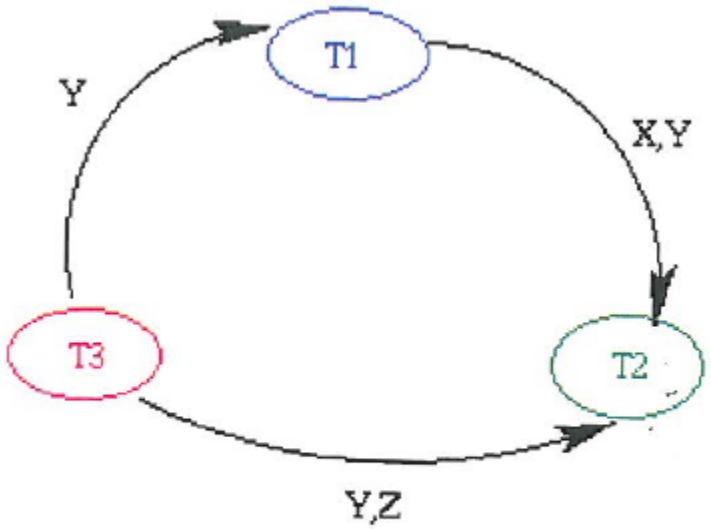
T1 ecrit avant T2 donc T1 precede T2.

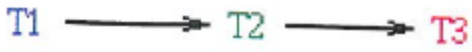
Pour l'objet Z:

Lecture de T3, Ecriture de T3, Lecture de T2.

Relations de precedence: T3 ecrit avant T2 donc T3 precede T2.

Le graphe de precedence ressemble alors a ce qui suit:





Conclusion: Il n'y a pas de cycle dans notre graphe de serialisabilite. On peut donc donner de cette execution une execution en serie equivalente:

Cela signifie que l'etat de la base de donnees (quelles que soient les valeurs des donnees) sera le meme apres l'execution en parrallele des 3 executions dans l'ordre donne qu'apres l'execution serie equivalente T3, T1, T2.