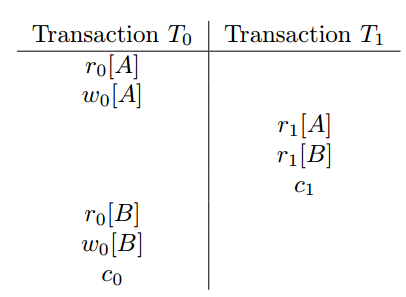
**Exercice 1 :**

Soit les trois transactions , et et les opérations de lecture () et d’écriture (*)*. On dénote respectivement par() et (), l’accès à la ressource par la transaction pour la lecture ou l’écriture.

1. Tester si un ordonnancement est sérialisable pour les scénarios suivants :
2. () ;() ;() ;() ;() ;
3. () ;() ;() ;() ;() ;
4. () ;() ;() ;() ;() ;

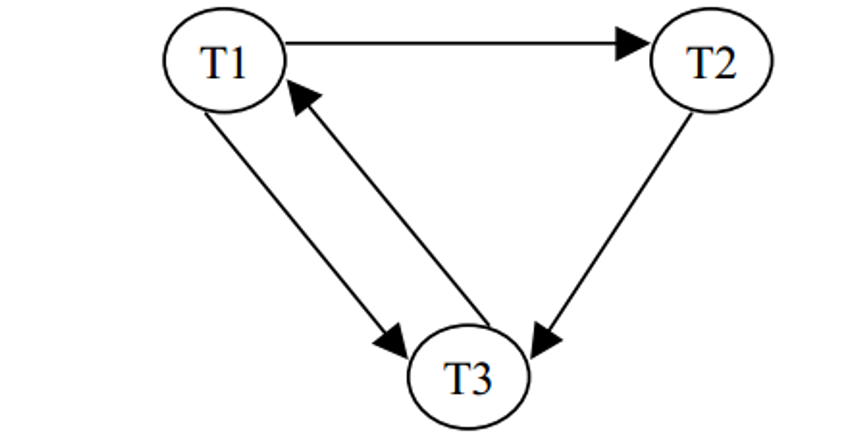
Considérez les deux transactions et l’ordonnancement suivants :

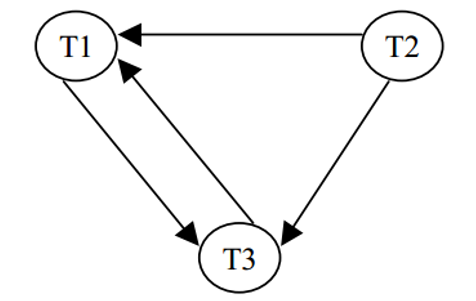


On dénote respectivement par() et (), l’accès à la ressource par la transaction pour la lecture ou l’écriture.

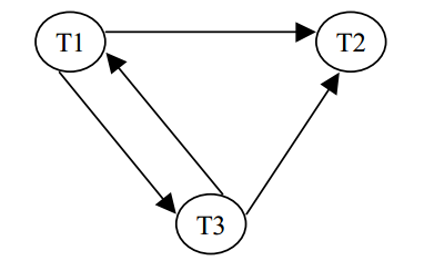
1. Cet ordonnancement est sérialisable par conflit ? Expliquez ?
2. Montrez comment l’algorithme 2PL (*two phase locking*) peut garantir un ordonnancement sérialisable par conflit pour les mêmes transactions ci-dessus.

**Correction d’exercice 1 :**

 Non sérialisable



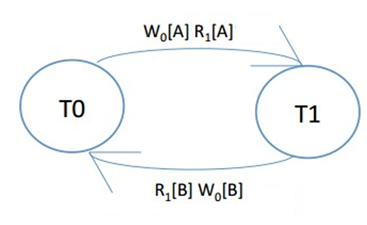
Non sérialisable



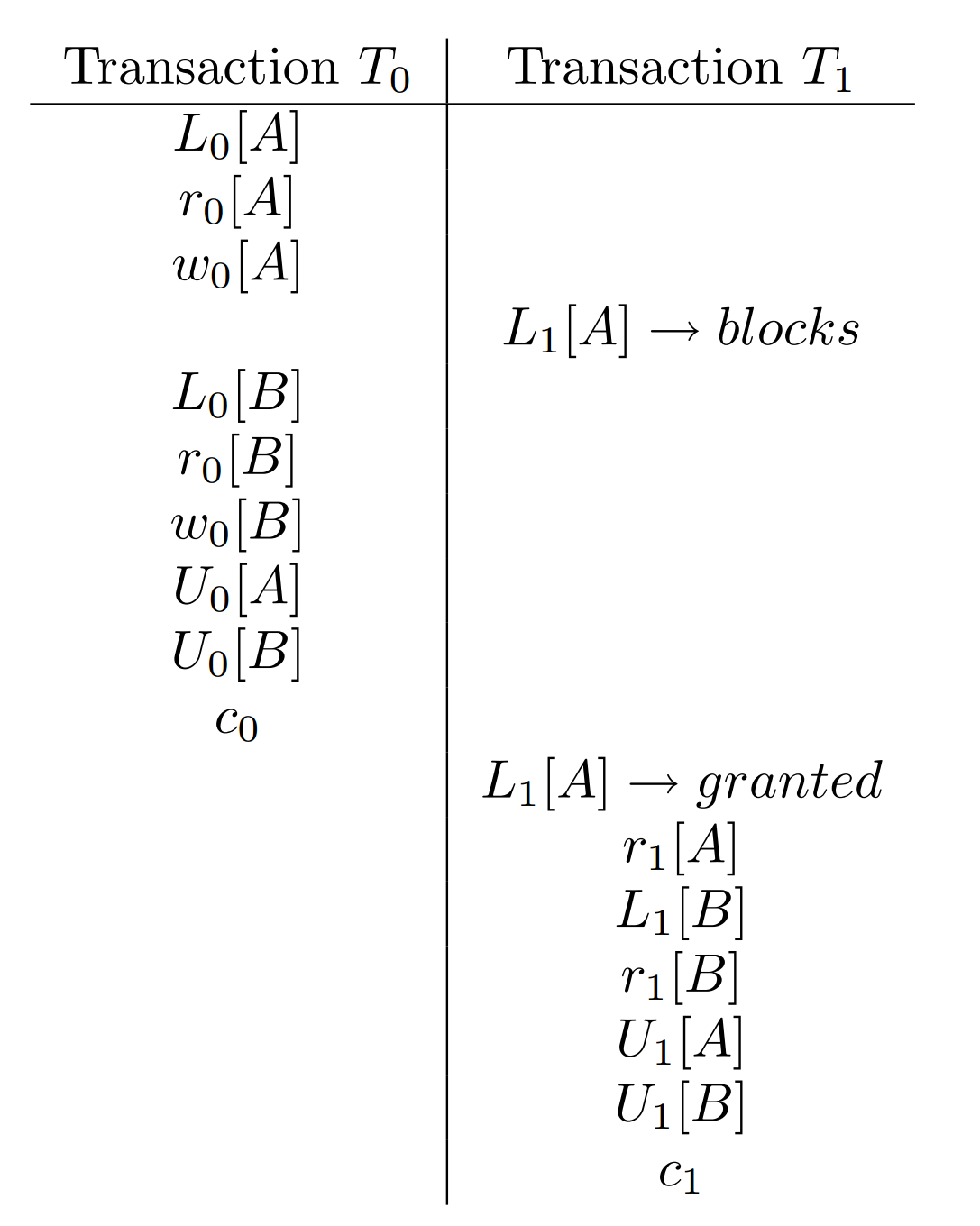
Non sérialisable

1) L’ordonnancement **n'est pas sérialisable** **par conflit** car le graphe de précédence contient un cycle.

* Le graphe a une arête T0 → T1 car l’ordonnancement contient w0[A] → r1 [A].
* Le graphe a une arête T1 → T0 car l’ordonnancement contient r1 [B] → w0 [B].



1. Montrez comment l’algorithme 2PL (*two phase locking*) peut garantir un ordonnancement sérialisable par conflit pour les mêmes transactions ci-dessus. **(L :Lock, U : Unlock)**



**Exercice 2: Gestion des transaction**

1. Un système de traitement des transactions ne connaît jamais d'échecs, la journalisation (par ex. *redo logging*) est complètement inutile ? **🞎** Vrai, **🞎** Faux , pourquoi!!
2. Quels sont les 5 états d'une transaction ?
3. Quelles propriétés ACID sont garanties par le système de reprise (*recovery system*) ?
4. Expliquez la différence entre la reprise à chaud/ et la reprise à froid ?
5. Expliquer l’intérêt des deux listes UNDO et REDO ?
6. Quelles sont les données validées après l’exécution des opérations suivantes ?

**A**)

SET AUTOCOMMIT =0 ;

INSERT INTO Coordonnées(X,Y) values (5, 6);

SAVEPOINT my\_savepoint\_1;

INSERT INTO Coordonnées(X,Y) values (7, 8);

SAVEPOINT my\_savepoint\_2;

INSERT INTO Coordonnées(X,Y) values (9, 10);

ROLLBACK TO my\_savepoint\_1;

INSERT INTO Coordonnées(X,Y) values (11, 12);

INSERT INTO Coordonnées(X,Y) values (23, 6);

**B)**

SET AUTOCOMMIT=0;

START TRANSACTION;

SAVEPOINT sp1;

INSERT INTO villes(cp, nom, adresse) VALUES('14000','nom1', ‘adresse1);

SAVEPOINT sp2;

INSERT INTO villes(cp, nom, adresse) VALUES('14002','nom2', ' adresse2');

ROLLBACK TO SAVEPOINT sp2;

COMMIT;

SELECT \* FROM villes;

**Exercice Correction 2 :** Quelles sont les données validées après l’exécution des opérations suivantes? **.**

SET AUTOCOMMIT =0 ;

INSERT INTO Coordonnées(X,Y) values (5, 6);

SAVEPOINT my\_savepoint\_1;

INSERT INTO Coordonnées(X,Y) values (7, 8);

SAVEPOINT my\_savepoint\_2;

INSERT INTO Coordonnées(X,Y) values (9, 10);

ROLLBACK TO my\_savepoint\_1;

INSERT INTO Coordonnées(X,Y) values (11, 12);

INSERT INTO Coordonnées(X,Y) values (23, 6);

Résultat vide

Transactions pas encore validées (coMMiTées)

|  |  |
| --- | --- |
| **X** | **Y** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

SET AUTOCOMMIT=0;

START TRANSACTION;

SAVEPOINT sp1;

INSERT INTO villes(cp, nom, adresse) VALUES('14000','nom1', ‘adresse1);

SAVEPOINT sp2;

INSERT INTO villes(cp, nom, ville) VALUES('14002','nom2', ' adresse2');

ROLLBACK TO SAVEPOINT sp2;

COMMIT;

SELECT \* FROM villes;

|  |  |  |
| --- | --- | --- |
| **cp** | **nom** | **adresse** |
| 14000 | Nom1 | Adresse1 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**Exercice 3: l'estimation des coûts**

On considère deux relations R(A,B,C,D) et S(D,E) avec les statistiques suivantes :

T(R )=100, V(R, A) =100, V(R,B)=10, V(R,C)=1, V(R,D)=50, T(S)=500, V(S,D) =30 ,

V(S,E) =100, B(R) = 50 et B(S )= 100.

On dénote par :

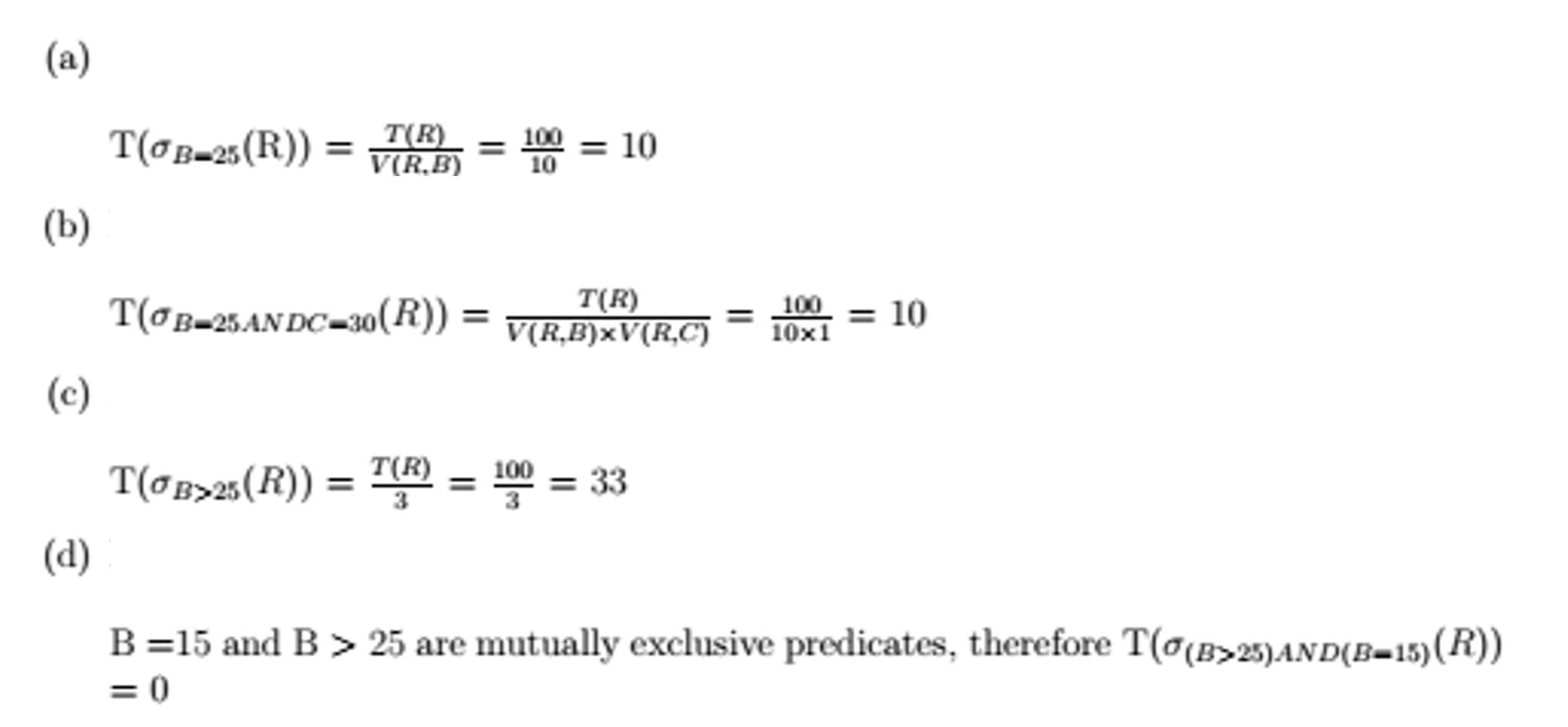
* V(X, Y): le nombre de valeurs distinctes de l’attribut Y de la relation X ,
* T(Y): Le nombre de tuples de la relation Y ,
* B(Y) : le nombre de blocks pour stocker la relation Y.

On suppose que les données suivent une distribution uniforme, estimer le nombre de tuples satisfaisant les prédicats suivants :

*Calculer le nombre d'opérations d'entrée/sortie de disque pour les deux opérations de jointure suivantes :*

1. avec l’utilisation de l'algorithme de jointure par hachage*.*
2. avec l’utilisation de l'algorithme de jointure des boucles imbriquée et *101 pages mémoire sont disponibles.*

**Correction Exercice 3 : l'estimation des coûts**



*(e ) Jointure par hachage -> = 3 x (B(R) + B(S)) = 3X (50+100)=450*

*(f ) Jointure par deux boucles imbriquées*

*= B(R )+ (T(R )\* B(S) / pages-1)= 50 + (*100 x 100)/(101-1)=150

**Exercice 5:** (5 points)

Chaque groupe de mots ci-dessous appartient à une catégorie. A vous de trouver la catégorie en essayant d’être le plus précis que possible. La première ligne est un exemple.

|  |  |  |
| --- | --- | --- |
|  |  | **Catégorie** |
|  | Oracle, DB2, SQL Server, PostgreSQL | SGBD Relationnels |
|  |  | |
| 1 | COMMIT, ROLLBACK, SAVEPOINT, GRANT, REVOKE |  |
| 2 | Haute Disponibilité, Scalabilité,SQL Support, ACID |  |
| 3 | Photocopie, Copie vivante, Fragmentation |  |
| 4 | Haute disponibilité, Scalabilité, Pas de ACID |  |
| 5 | Entité/Association (EA), UML, Express. |  |
| 6 | Relationnel, Multidimensionnel, Réseau, Relationnel-Objet |  |
| 7 | Clé-valeur, Graphes, Documents, Colonnes |  |
| 8 | Shared memory, Shared disks, Shared nothing |  |

**Correction Exercice 5 : (5 points)** Chaque groupe de mots ci-dessous appartient à une catégorie. A vous de trouver la catégorie en essayant d’être le plus précis que possible. La première ligne est un exemple.

|  |  |  |
| --- | --- | --- |
|  |  | **Catégorie** |
|  | Oracle, DB2, SQL Server, PostgreSQL | SGBD Relationnels |
|  |  | |
| 1 | COMMIT, ROLLBACK, SAVEPOINT, GRANT, REVOKE | Contrôle de transaction, LCD : langage de contrôle de données (**1)** |
| 2 | Haute Disponibilité, Scalabilité,SQL Support, ACID | NewSQL (**0,5)** |
| 3 | Photocopie, Copie vivante, Fragmentation | Modèle de distribution de BD (ou base de données repartie) (**0,5)** |
| 4 | Haute disponibilité, Scalabilité, Pas de ACID | NoSQL(**0,5)** |
| 5 | Entité/Association (EA), UML, Express. | MCD (Modèle conceptuel des données) **(1)** |
| 6 | Relationnel, Multidimensionnel, Réseau, Relationnel-Objet | MLD Modèle logique des données(**0,5)** |

**Exercice 6:**

Utiliser l’optimisation à base de règles (RBO : Rule-Base Optimisation) pour trouver un arbre optimal de la requête SQL suivante? **(4 pt).**

(SELECT DISTINCT Pnumber

FROM PROJECT, DEPARTMENT, EMPLOYEE

WHERE Dnum = Dnumber AND Mgr\_ssn = Ssn

AND Lname = ‘Smith’)

UNION

(SELECT DISTINCT Pnumber

FROM PROJECT, WORKS\_ON, EMPLOYEE

WHERE Pnumber = Pno AND Essn = Ssn

AND Lname = ‘Smith’)

