USTHB
Faculté d'Informatique 2023/2024

**Examen Administration des SGBD**

**M1 : IL-SII**

**Enoncé :**

La satisfaction client est une préoccupation majeure pour un opérateur mobile. Chaque opérateur vise à offrir les meilleurs services afin de retenir et fidéliser les clients existants et attirer de nouveaux clients potentiels. Généralement, les opérateurs créent des enquêtes pour mesurer la satisfaction. Le schéma relationnel suivant décrit la BD relative à l'enquête.

BUs (idbu, BU)

Wilayas (idwilaya, nomwilaya, *idbu*)

Clients (idclient, nom, num_tel, dateactivation, profil_global, type, statut, tech_type, *idwilaya*)

Questions (idquestion, question)

Réponses (*idquestion, idclient,* date_réponse, réponse)

En gras et souligné les clés primaires, en italique les clés étrangères

Nous disposons des informations suivantes

| | |
|---|---|
| BUs | Card(BUs)=4 |
| Wilayas | Card(Wilayas)=58 |
| Clients | Card(Clients)= 84000 |
| Questions | Card(Questions)=14 |
| Réponses | Card(Réponses)= 588000 |
| La page d'entrée/Sortie i.e. page d'échange entre la mémoire centrale et secondaire peut contenir 7 enregistrements (tuples). Le temps de chargement d'une page E/S est 1 milliseconde. On suppose que chaque attribut est codé sur 50 octets. | |

- BUs : Business Unit, cette table correspond à une répartition de wilayas dans des régions, suivant un objectif commercial. Il existe 4 BU (Centre, Est, Ouest, Sud).
- Le nombre de wilayas est 58.
- type : on distingue trois types prépayé, post payé et hybride.
- statut : cette colonne donne l'information sur le statut du client actif ou terminé.
- profil_global: cette colonne indique l'offre activée sur la puce du client. Il existe 30 profils.
- tech_type: cette colonne contient le type de la technologie de la puce du client (2G, 3G, 4G et 5G).
- Réponse est un entier qui appartient à la liste {0,1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

**Exercice 1 : Fonctions générales de SGBD et Optimisation (12 points)**

**1.** L'administrateur exécute les requêtes suivantes :

| N° Ligne de code SQL | Code SQL |
|---|---|
| 1<br>2<br>3<br>4<br>5<br>6 | create profile Mobile_Profile<br>limit<br>sessions_per_user 3<br>cpu_per_call 1000<br>logical_reads_per_session 24000 ;<br>Alter user ENQUETEUR profile Mobile_Profile ; |

**a.** Décrire chaque ligne de code SQL. 01 pts

Réponse :

| N° Ligne de code SQL | Interprétation |
|---|---|
| 1 | Création d'un profil nommé Mobile_Profile |
| 2 | Pour limiter les ressources |
| 3 | nombre de sessions concurrentes autorisées est 3 |

| | |
|---|---|
| 4 | temps CPU autorisé pour un appel noyau est 10 secondes |
| 5 | le nombre maximal de bloc lus durant une session est 24000 |
| 6 | Attribuer le profil Mobile_Profile à l'utilisateur ENQUETEUR |

**b.** L'utilisateur ENQUETEUR reçoit les privilèges nécessaires pour se connecter et manipuler la BD. ENQUETEUR exécute les requêtes ci-dessous en séquence ; Pour chaque requête, préciser si elle s'exécute correctement en considérant les limites du profil Mobile_Profile. 01 pts

| N ° de la requête | Requête SQL |
|---|---|
| 1 | Select * from wilayas ; |
| 2 | Select * from clients ; |

Réponse :

| N ° de la requête | Réponse du SGBD |
|---|---|
| 1 | Card(Wilayas)=58<br>Nombre de bloc : 58/7= 9 <24000<br>Tems d'exécution : 9*1 ms<10 s<br>La requête respecte les limites de profil donc la requête s'exécute avec succès |
| 2 | Card(Clients)= 84000<br>Nombre de bloc : 84000/7= 12000 +9 (R1)<24000<br>Tems d'exécution : 12000*1 ms=12 secondes>10 secondes<br>La requête ne respecte pas les limites de profil donc la requête ne s'exécute pas avec succès. |

**2.** Les tables BUs et Wilayas sont déjà créées. Donner le script SQL pour créer la table Clients avec toutes les contraintes d'intégrité. 1.5 pts

Réponse :

```
create table Client  (idclient integer, nom varchar2(100), num_tel varchar2(12), dateactivation date,
profil_global integer, type varchar2(15), statut varchar2(10), tech_type varchar2(2), idwilaya integer,
constraint pk_ Client  primary key(idclient),
constraint fk_wilaya_ client foreign key(idwilaya)  references wilayas (idwilaya) on delete cascade,
constraint ck_ type check (type in ('prépayé',' post payé', 'hybride')),
constraint ck_ statut check (statut in ('actif','terminé')),
constraint ck_ tech_type check (tech_type in ('2G', '3G',' 4G', ' 5G')),
constraint ck_ profil_global check (profil_global>=1 and profil_global<=30)));
```

**3.** Le Net Promoter Score ou NPS est un indicateur de la satisfaction client. Il est basé sur une question simple : « Sur une échelle de 0 à 10, recommanderiez-vous notre entreprise/nos services/nos produits à un de vos proches ? ». Les répondants sont ensuite classés en trois catégories:

- Les promoteurs : Ce sont ceux qui ont répondu 9 ou 10.
- Les passifs : Ce sont ceux qui ont répondu 7 ou 8.
- Les détracteurs : Ce sont ceux qui ont répondu de 0 à 6.

Le score NPS se calcule en soustrayant le pourcentage de détracteurs du pourcentage de promoteurs.

Le NPS est interprété comme suit :

- Un score NPS nul indique une satisfaction neutre
- Un score NPS positif (supérieur à 0) est considéré comme bon.
- Un score NPS négatif indique une insatisfaction des clients.

**a.** Définir une vue qui permet de sauvegarder les réponses des clients qui ont répondu à la question relative à NPS. Donner les effets sur les catalogues. 01.5 pts

Réponse :

```
create view NPS (idclient, reponse) as
 (select idclient, réponse
from reponses R, questions Q
where question='Sur une échelle de 0 à 10, recommanderiez-vous notre entreprise/nos services/nos produits à un de vos proches ?'
and R. idquestion=Q. idquestion );
effets sur les catalogues:
Catalogue des vues: ajout d'une ligne qui décrit la vue NPS.
```

**b.** Ecrire une procédure qui calcule le taux de promoteurs, taux de passifs, taux de détracteurs ainsi que le NPS et l'interprète. 01.5 pts

Réponse :

```
create or replace procedure Calcul_NPS as
all_response, promoteurs, passives, détracteurs integer ;
NPS number ;
begin
select count(*)  into all_response from NPS;
select count(*)  into promoteurs from NPS where réponse=9 or réponse=10;
select count(*)  into passifs from NPS where réponse=7 or réponse=8;
select count(*)  into détracteurs from NPS where réponse>=0 or réponse<=6;
dbms_output.put_line('Le taux de promoteurs est '|| round(promoteurs/ all_response) ;
dbms_output.put_line('Le taux de passifs est '|| round(passifs/ all_response) ;
dbms_output.put_line('Le taux de détracteurs est '|| round(détracteurs / all_response) ;
NPS := round((promoteurs- détracteurs) / all_response) ;
dbms_output.put_line('Le NPS est '||NPS) ;
case
when (NPS=0) then dbms_output.put_line('Satisfaction neutre') ;
when (NPS>0) then dbms_output.put_line('Une bonne satisfaction') ;
when (NPS<0) then dbms_output.put_line('Insatisfaction') ;
end case ;
end Calcul_NPS;
/
```

**4.** Le responsable de l'enquête veut avoir le nombre de clients ayant répondu à chaque question sans avoir à le calculer à chaque fois.

**a.** Que faut-il faire ? Donner les commandes SQL associées. 01 pts

Réponse :

- Ajouter un attribut nb_clients dans la table Questions et définir un trigger « maj_nb_reponse_question_trigger » qui mit à jour (incrémente ou décrémente) l'attribut nb_clients à chaque ajout, modification ou suppression dans la table Réponses.
- Alter table Question add nb_clients integer default 0;

**b.** Ecrire le script SQL permettant de faire automatiquement ce calcul sachant que la BD ne contient pas de tuples. 01 pts

Réponse :

```
create or replace trigger « maj_nb_reponse_question_trigger
after
insert or delete or update of(idquestion)
on réponses
for each ROW
begin CASE
   WHEN INSERTING THEN  update questions set clients = clients +1  where idquestion = :new.idquestion;

   WHEN DELETING THEN update questions set clients = clients -1  where idquestion = :old.idquestion;
   WHEN UPDATING THEN
                          Begin
                          update questions set clients = clients +1  where idquestion = :new.idquestion;
                          update questions set clients = clients -1  where idquestion = :old.idquestion;
                         End;
END CASE;
EXCEPTION
WHEN OTHERS then  DBMS_OUTPUT.PUT_LINE('error : '||sqlcode||' '||sqlerrm);
END;
/
```

**5.** Le responsable de l'enquête veut afficher la liste de clients actifs (nom, num_tel, dateactivation) relatifs aux deux BU Centre et Ouest.

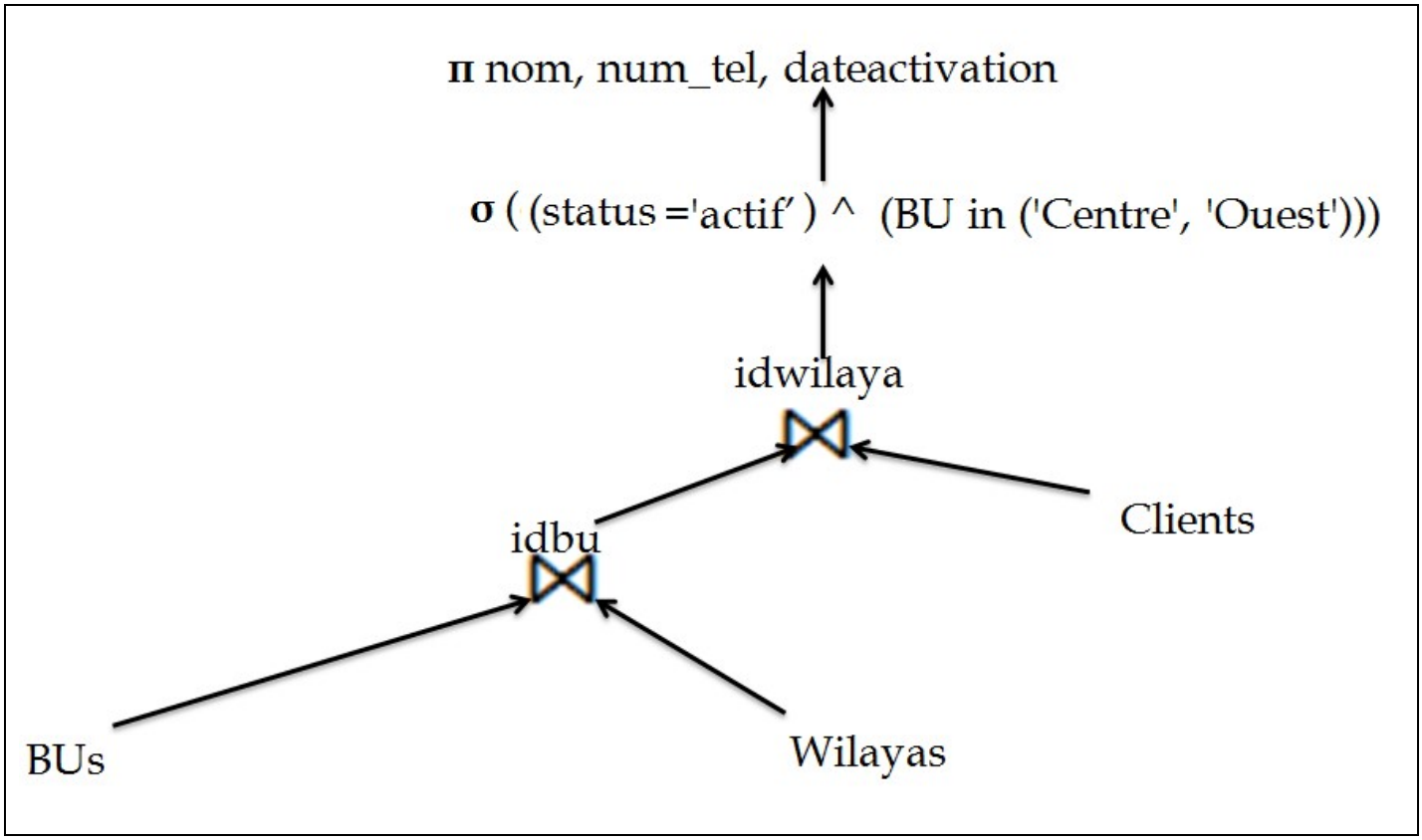
**a.** Donner la requête SQL. 01 pts

Réponse :

```
Select nom, num_tel, dateactivation
From BUs B, Wilayas W, Client C
Where status='actif'
And BU in ('Centre', 'Ouest')
And B.idbu=W.idbu
And W.idwilaya=C.idwilaya;
```

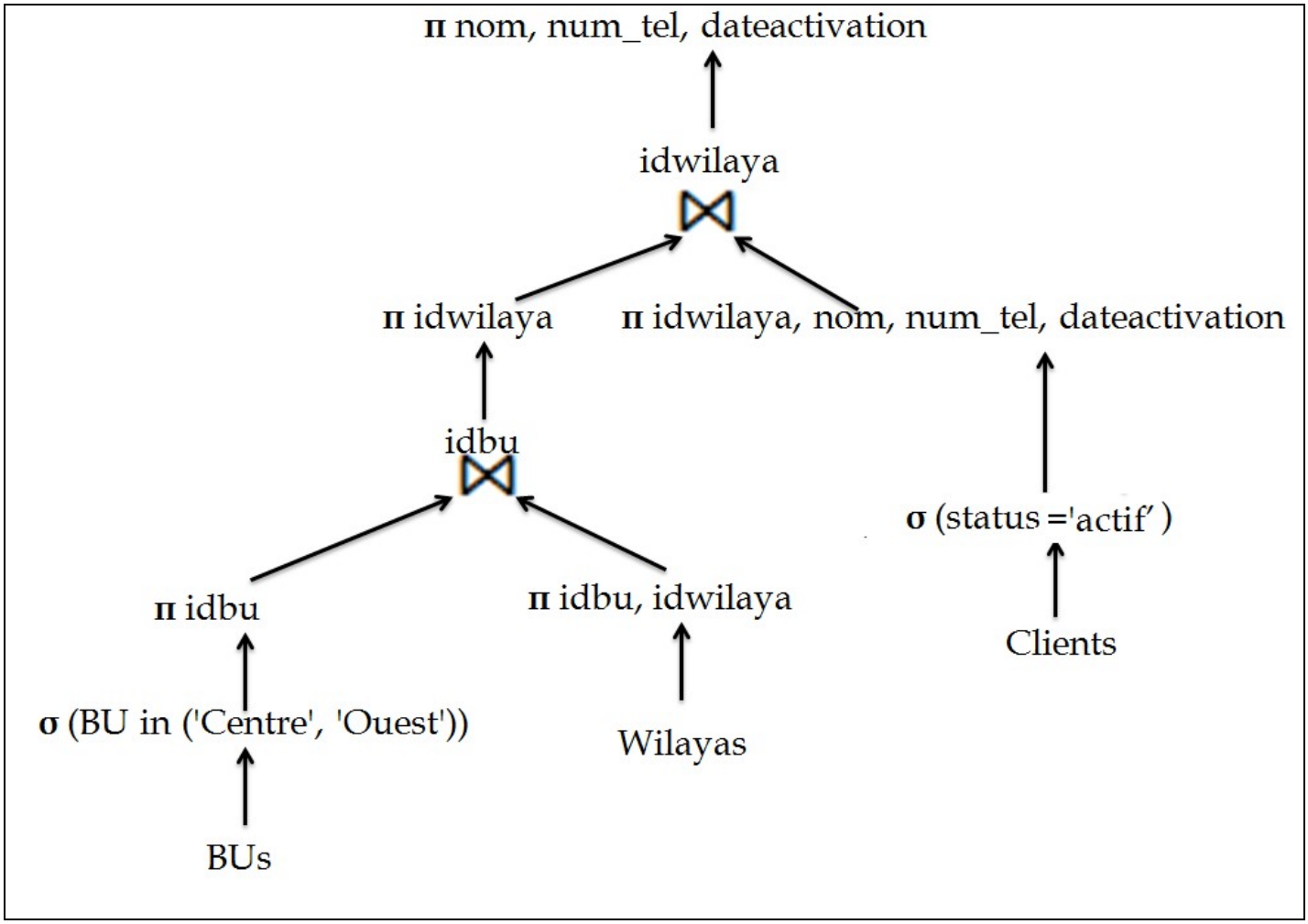
**b.** Donner un arbre algébrique non optimisé. 0.5 pts

Réponse :

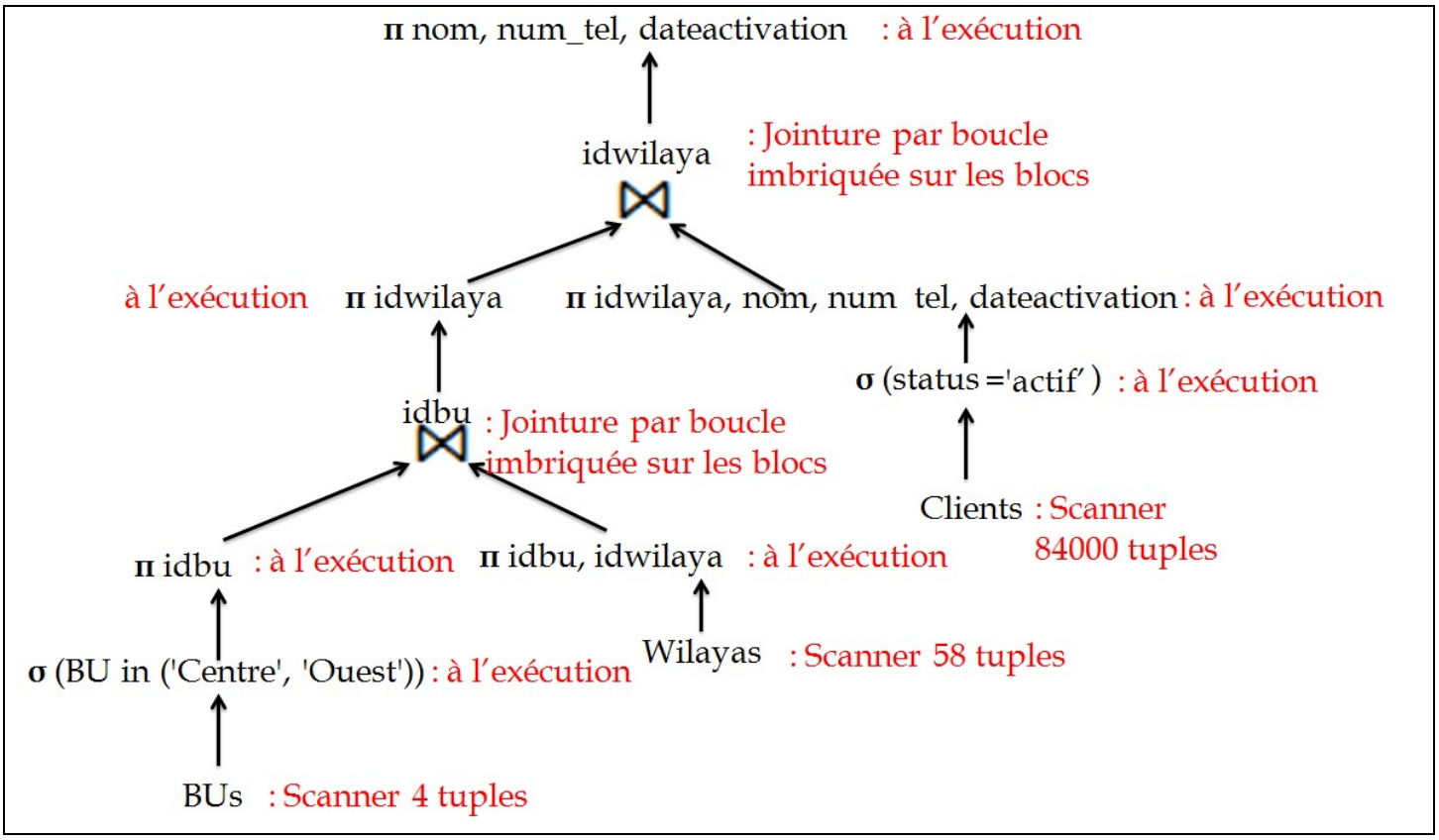



**c.** Générer un arbre optimisé en utilisant les règles de transformation. 0.5 pts

Réponse :



**d.** Estimer le coût d'exécution de la requête selon l'arbre optimisé sachant que les résultats intermédiaires ne peuvent pas être stockés dans la RAM. 03.5pts




| Opération | Coût | Cardinalité |
|---|---|---|
| **R1=** σ (BU in ('Centre', 'Ouest'))(BUs) | Charger toute la table BUs = 4/7= 1 page.<br>Chaque page peut tenir 7 tuples. | -Card(BUs)*(1/val(BU)*card(('Centre', 'Ouest'))<br>-4*(1/4)*2=2 tuples |
| Sauvegarde du résultat **R1** en MS | 2/5 = 1 page | |
| **R2=** R1 ⋈ Wilayas | 1+1*(58/7) = 10 pages(boucles imbriquées) | Jointure sur la clé étrangère<br>2 tuples de Resultat R1 avec 58 tuples de wilayas<br>Nombre de tuples 29 |
| Sauvegarde du résultat **R2** en MS | 29/7 = 5 pages | |
| R3= σ (statut='actif' )(Clients)<br><br>**R4=** R2 ⋈ R3 | Charger toute la table Client= 84000/7= 12000 pages<br>Sauvegarder le résultat de R3 Card(R3) en MS<br>42000/7=6000 pages<br>5+5*(6000) = 30005 pages(boucles imbriquées) | Card(R3)=84000/2=42000 tuples<br>Jointure sur la clé étrangère<br>29 tuples de Resultat R2 avec 42000 tuples de R3<br>Nombre de tuples 21000 |
| π | À l'exécution | 21000* [size (nom)+ size (num_tel)+size (dateactivation) ]<br>21000*3*50<br>3150000 octets |
| **Total** | 48022 pages | 3150000 octets |

**Exercice 2 : Gestions des accès concurrents et reprise après panne (08 points)**

1. Dans un ordonnancement O sur un ensemble de transactions {T1 ; … ; Tn}, on dit que deux transactions Ti ; Tj partagent un granule X de la base de données si elles exécutent toutes les deux une action sur ce granule.

Un ordonnancement O est dit "Chary" si :

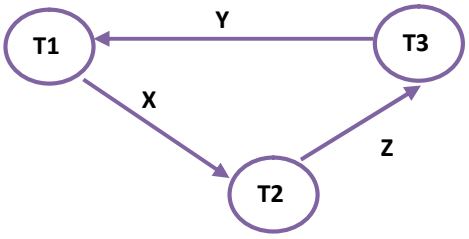
- Aucune transaction n'utilise deux fois le même granule.
- Pour chaque Ti ; Tj ∈ {T1 ; … ; Tn}, Ti et Tj partagent au plus un granule.

Prouver ou réfuter les affirmations suivantes :

**a.** Tout ordonnancement "Chary" est sérialisable. 01 pts

L'affirmation peut être réfutée par le contre-exemple **O1** suivant : **R1(X) W2(X) R3(Y) W1(Y) R2(Z) W3(Z)**

1. **O1 est chary vérifie les deux conditions ci-dessus**
2. **Graphe de précédence de «O1 »**




Les conflits :

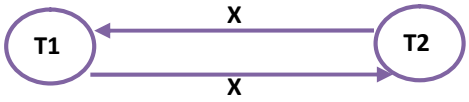
- Sur X : R1(X)-W2(X);
- Sur Y : R3(Y)- W1(Y) ;
- Sur Z : R2(Z)-W3(Z) ;

**O1 est chary mais le graphe de précédence porte un cycle d'où O1 n'est pas sérialisable.**

**b.** Tout ordonnancement "Chary" sur deux transactions est sérialisable. 01 pts

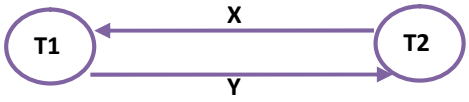
Nous prouvons cette affirmation en montrant que tout ordonnancement qui n'est pas sérialisable n'est pas « Chary ».
Considérons un ordonnancement O sur deux transactions {T1, T2} et supposons que O n'est pas sérialisable ce qui signifie qu'il y a un cycle dans le graphe de précédence associé à O. cela signifie qu'il y a au moins deux paires différentes d'actions conflictuelles dans O, disons $\alpha(X), \beta(X)$ et $\gamma(Y), \delta(Y)$et $\alpha, \beta, \gamma, \delta$ peuvent être lecture ou écriture de T1 ou T2. Il y a deux cas X=Y, soit X<>Y.

Cas X=Y :




nous avons au moins 3 actions qui utilisent le même granule et donc au moins l'une des deux transactions utilise le même granule deux fois ce qui contredit la première condition d'un ordonnancement « Chary ».

Cas X<>Y :




T1 et T2 partagent deux éléments différents ce qui contredit la deuxième condition d'un ordonnancement « Chary ».

**c.** Tout ordonnancement "Chary" sur deux transactions est un ordonnancement sérialisable par V2P. 01pts

L'ordonnancement O est « Chary » donc les deux transactions partagent un seul granule et chaque transaction utilise ce granule une seule fois soit en lecture ou en écriture. Soit X le granule partagé entre T1 et T2.

**Cas 1 :** T1 a verrouillé le granule X avant T2, donc toute écriture de T2 ne sera faite que lors que T1 termine et libère le verrou sur X et réveille T2 donc pas d'inter blocage.

**Cas 2 :** T2 a verrouillé le granule X avant T1, donc toute écriture de T1 ne sera faite que lors que T2

termine et libère le verrou sur X et réveille T1 donc pas d'inter blocage.

**2.** Considérons l'ordonnancement O des transactions T1, T2, T3 et T4

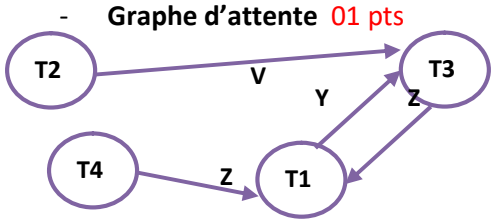
O : R1(Z) W3(Y) W3(V) R1(Y) R2(V) W2(Y) W3(X) R2(X) R2(Z) W3(Z) W4(Z) W4(X) W2(X)

**a.** Donner le scenario d'exécution de cet ordonnancement dans le cas du verrouillage à deux phases. Construire le graphe d'attente de cet ordonnancement. Existe-t-il un deadlock ? Si oui, proposer une solution à ce problème. 03.5 pts

- **V2P** 01.5 pts

| **Transaction** | **Action** | **Demande de verrous** | **Réponse** |
|---|---|---|---|
| T1 | R1(Z) | SLOCK(Z) | OK |
| T3 | W3(Y) | XLOCK(Y) | OK |
| T3 | W3(V) | XLOCK(V) | OK |
| T1 | R1(Y) | SLOCK(Y) | Non, T1 attend la fin de T3 |
| T2 | R2(V) | SLOCK(V) | Non, T2 attend la fin de T3 |
| T2 | W2(Y) | XLOCK(Y) | T2 est en attente |
| T3 | W3(X) | XLOCK(X) | OK |
| T2 | R2(X) | SLOCK(X) | T2 est en attente |
| T2 | R2(Z) | SLOCK(Z) | T2 est en attente |
| T3 | W3(Z) | SLOCK(Z) | Non, T3 attend la fin de T1 |
| T4 | W4(Z) | XLOCK(Z) | Non, T4 attend la fin de T1 |
| T4 | W4(X) | XLCOCK(X) | T4 est en attente |
| T2 | W2(X) | XLOCK(X) | T2 est en attente |

- **Graphe d'attente** 01 pts




- De graphe d'attente, on constate qu'on a le cycle {T1, T3, T1} d'où on a un inter-blockage. Pour lever l'inter blocage on choisit une transaction victime parmi {T1, T3} on l'annule et on la relance à la fin. La transaction T1 est la victime (n'a pas fait de mise à jour).

Rollback(T1): UNLOCK(Z) et réveiller T3 01 pts

| | | | |
|---|---|---|---|
| T3 | W3(Z) | XLOCK(Z) | OK; Fin de T3; Unlock(Y) ; Unlock(V), Unlock(X) ; Unlock(Z) Réveiller T2 |
| T2 | R2(V) | SLOCK(V) | OK ; |
| T2 | W2(Y) | XLOCK(Y) | OK |
| T2 | R2(X) | SLOCK(X) | OK |
| T2 | R2(Z) | SLOCK(Z) | OK |
| T2 | W2(X) | XLOCK(X) | Fin de T2 ; Unlock(X), Unlock(Z), Unlock(Y) ; Unlock(V) ; Réveiller T4 |
| T4 | W4(Z) | XLOCK(Z) | OK ; |
| T4 | W4(X) | XLCOCK(X) | OK ; Fin de T4 ; Unlock(X), Unlock(Z) ; |
| **Relancer T1 dés son début :** | | | |
| T1 | R1(Z) | SLOCK(Z) | OK |
| T1 | R1(Y) | SLOCK(Y) | OK ; Fin de T1; Unlock(Z), Unlock(Y) ; |

L'exécution série équivalente: **T3 T2 T4 T1** après le levé d'inter-blocage.

**3.** Soient cinq transactions T1, T2, T3, T4 et T5. Nous supposons le contenu suivant du journal de transaction : 01.5 pts

| N° de la ligne dans le log | Instruction dans le log |
|---|---|
| 1 | < START T1 > |
| 2 | < T1, A, 5, 10 > |
| 3 | < START T2 > |
| 4 | < T2, B, 0, 5 > |
| 5 | < T1, C, 3, 7 > |
| 6 | < START T3 > |
| 7 | < T3, D, 4, 12 > |
| 8 | < COMMIT T1 > |
| 9 | < CHECKPOINT 1> |
| 10 | < START T4 > |
| 11 | < T2, E, 10, 5 > |
| 12 | < COMMIT T2 > |
| 13 | < T3, F, 20, 1 > |
| 14 | < T4, G, 5, 15 > |
| 15 | < COMMIT T3 > |
| 16 | < START T5 > |
| 17 | < T5, H, 0, 3 > |
| 18 | < CHECKPOINT 2> |
| 19 | < COMMIT T5 > |
| | *C*R*A*S*H* |

Quelles sont les valeurs des granules après la récupération ? Justifier.

Etats des transactions avant la panne 0.5 pts
T1, T2 et T3 sont validées terminées.
T5 est partialement validée.
T4 est active.
Après récupération 0.5 pts
T1, T2 et T3 rien à faire.
T5 refaire les mises à jour après CHECKPOINT2 (pas de MAJ faite par T5).
T4 défaire les mises à jour effectuées en CHECKPOINT2
**d'où l'état suivant des granules :**
(A, B, C, D, E, F, G, H)= (10, 5, 7, 12, 10, 1, 5,3) 0.5 pts

Bon courage