

**Examen Administration des SGBD  
M1 : MIV**

**Enonce :**

Une société de distribution d'électricité gère, à travers des agences, un ensemble de clients suivant deux types : les particuliers et les entreprises. Les clients ont des activités professionnelles (par exemple « commerce », « usine », « service » etc.). Des contrats sont élaborés en fonction des clients et des factures sont émises régulièrement suivant leur consommation. Une division du territoire suivant des régions permet d'en simplifier la gestion. Soit le schéma simplifié de cette étude cas :

Agence ( **NumA**, LibelleA, NumZone, AdresseA)

Zone (**NumZ**, Wilaya, Superficie)

Client (**NumCli**, NomCli, AdresseCli, TypeCli, Code\_Activite, NumA)

Activite( **Code Activite**, Libelle\_Activite)

Contrat ( **NumC**, Tarif, Date\_creation, Date\_Effet, Date\_Fin,NumCli)

Facture ( **NumFact**, NumC, DateFact, Periode, Consommation, Montant)

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

Nous disposons les informations suivantes

Agence	Card(Agence)=5800
Zone	Card(Zone)=232, Val (wilaya)=58
Client	Card(Client)= 23200
Activite	Card(Activite)=464
Contrat	Card(contrat)=46400
Facture	Card(facture)=92800
La page d'entrée/Sortie i.e. page d'échange entre la mémoire centrale et secondaire peut contenir 116 enregistrements (tuples).	

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

1. La base de données possède deux utilisateurs, un utilisateur DBA qui a tous les privilèges possibles et propriétaire des tables du schéma relationnel précédent. Un utilisateur ADMINIELEC qui gère les clients, les activités, les contrats et factures en termes d'ajout, de modification et de suppression. Les utilisateurs partagent les deux tablespaces DBAELEC\_TBS et DBAELEC\_TEMP\_TBS avec un quota illimité.

a. Créer les deux utilisateurs avec leurs privilèges.

```
create user DBA identified by psw
default tablespace DBAELEC_TBS
quota unlimited on DBAELEC_TBS
temporary tablespace DBAELEC_TEMP_TBS;
grant all privileges to DBA ;

create user ADMINIELEC identified by psw
default tablespace DBAELEC_TBS
quota unlimited on DBAELEC_TBS
temporary tablespace DBAELEC_TEMP_TBS;
grant insert, update, delete on client to ADMINIELEC;
grant insert, update, delete on activite to ADMINIELEC;
grant insert, update, delete on contrat to ADMINIELEC;
grant insert, update, delete on facture to ADMINIELEC;
```

b. Si on veut rendre la table facture visible en lecture par les clients de l'entreprise. Que faut-il faire ? Quel est l'utilisateur qui peut exécuter cette requête ?

- L'utilisateur DBA car il a tous les privilèges.
- Grant select on client to public;

2. Donner le script SQL pour créer les tables Client et Facture.

```
CREATE TABLE Client (  
  NumCli integer,  
  NomCli varchar(30),  
  AdresseCli varchar(30),  
  TypeCli varchar(30),  
  Code_Activite varchar(10),  
  NumA integer,  
  constraint pk_client PRIMARY KEY (NumCli),  
  constraint fk_activite foreign key(Code_Activite) references Activite (Code_Activite) on delete cascade,  
  constraint fk_agence foreign key(NumA) references Agence(NumA) on delete cascade,  
  constraint ck_type check (TypeCli in ('particuliers','entreprise'))  
);
```

```
CREATE TABLE Facture (  
  NumFact integer,  
  NumC integer,  
  DateFact date,  
  Periode varchar(30),  
  Consommation number,  
  Montant number,  
  constraint pk_facture PRIMARY KEY (NumFact),  
  constraint fk_contrat foreign key(NumC) references Contrat (NumC) on delete cascade,  
);
```

3. L'administrateur veut, pour un besoin interne, avoir le total des gains pour chaque agence. Pour cela, il ajoute un attribut : **TOTAL\_GAINS**.

- a. Sur quelle table l'attribut est ajouté ? Donner le script SQL permettant d'ajouter cet attribut sachant que la BD contient déjà des tuples.

```
- L'attribut est ajouté à la table agence.  
- alter table Agence add TOTAL_GAINS NUMBER;  
- initialiser l'attribut TOTAL_GAINS par les valeurs qui existent dans la BD  
- create or replace procedure initialiser(NumAg Agence. NumA%type) is  
  totalgains number;  
- begin  
-   select sum(f.montant) into totalgains  
  from client c, contrat cc, facture f  
  where c.NumCli=cc.NumCli  
  and cc. Num=f. NumC  
  group by NumA;  
  update agence set TOTAL_GAINS = totalgains  
  where NumA=item. NumAg;  
- end initialiser;  
- /  
- declare  
-   cursor cr is select NumA from Agence;  
-   begin  
-     for item in cr  
-     loop  
-       initialiser(item. NumA);  
-     end loop;  
-   end;  
-   /
```

b. Ecrire le script permettant de mettre à jour automatiquement cet attribut.

```
create or replace trigger total_gains
after
insert or delete or update
on facture
for each ROW
BEGIN
CASE
WHEN INSERTING THEN update Agence set TOTAL_GAINS = TOTAL_GAINS +:new.Montant
                        where NumA=(select NumA from client where NumCli= +:new.NumCli) ;
WHEN DELETING THEN update Agence set TOTAL_GAINS = TOTAL_GAINS -:old.Montant
                        where NumA=(select NumA from client where NumCli= +:old.NumCli) ;;
WHEN UPDATING THEN update Agence set TOTAL_GAINS = TOTAL_GAINS +:new.Montant -:old.Montant
                        where NumA= NumA=(select NumA from client where NumCli= +:new.NumCli) ;;
END CASE;
EXCEPTION
WHEN OTHERS then DBMS_OUTPUT.PUT_LINE('error : '||sqlcode||' '||sqlerrm);
END;
/
```

4. L'administrateur veut afficher les entreprises (nomcli, adressecli) de la wilaya d'Alger.

a. Donner la requête SQL.

```
SELECT nomcli, adressecli
FROM client C, agence A, zone Z
WHERE C.NumA = A.NumA
AND A.NumZone=Z.NumZ
AND Z.Wilaya='Alger';
```

### Exercice 2 : Gestions des accès concurrents et reprise après panne 10 pts

Considérons les deux ordonnancements O1 et O2 des transactions T1, T2 et T3.

**O1 : R1(X) R2(Z) R1(Z) R3(X) R3(Y) W1(X) W3(Y) R2(Y) W2(Z) W2(Y)**

**O2 : R1(X) R2(Z) R3(X) R1(Z) R2(Y) R3(Y) W1(X) W2(Z) W3(Y) W2(Y)**

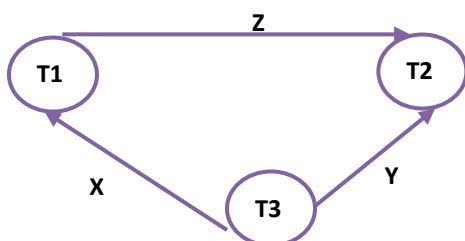
1. Vérifier si O1 et O2 sont sérialisables en identifiant les conflits et en construisant les graphes de précedence.

#### Ordonnancement O1

a. Les conflits :

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

b. Graphe de précedence



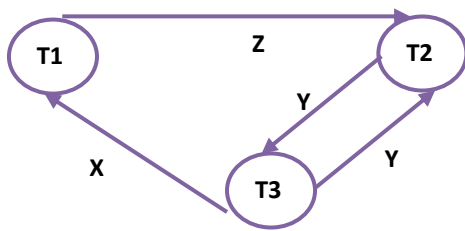
Le graphe ne contient pas de cycles ;  
D'où l'ordonnancement est sérialisable.

#### Ordonnancement O2 Les conflits :

- Sur X : R3(X)-W1(X);

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

a. Graphe de précédence



Le graphe contient plusieurs cycles par exemple {T3, T2, T3} ; d'où l'ordonnancement n'est pas sérialisable.

2. Appliquer le protocole de verrouillage à deux phases sur **O1** et **O2** jusqu'à l'exécution de toutes les actions. Quelle est l'exécution finale obtenue ?

- **V2P pour O1 :**

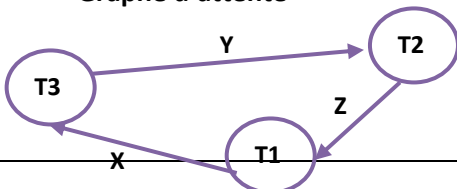
Transaction	Action	Demande de verrous	Réponse
T1	R1(X)	SLOCK(X)	OK
T2	R2(Z)	SLOCK(Z)	OK
T1	R1(Z)	SLOCK(Z)	OK
T3	R3(X)	SLOCK(X)	OK
T3	R3(Y)	SLOCK(Y)	OK
T1	W1(X)	XLOCK(X)	Non, T1 attend la fin de T3
T3	W3(Y)	XLOCK(Y)	OK ; Fin de T3 ; Unlock(X) ; Unlock(Y) ; Réveiller T1
T1	W1(X)	XLOCK(X)	OK, Fin de T1 ; Unlock(X) ; Unlock(Z) ;
T2	R2(Y)	SLCOCK(Y)	OK
T2	W2(Z)	XLOCK(Z)	OK
T2	W2(Y)	XLOCK(Y)	OK ; Fin de T2 ; Unlock(Y), Unlock(Z),

O1 s'exécute sans inter blocage ; L'exécution série équivalente: **T3 T1 T2**

- **V2P pour O2 :**

Transaction	Action	Demande de verrous	Réponse
T1	R1(X)	SLOCK(X)	OK
T2	R2(Z)	SLOCK(Z)	OK
T3	R3(X)	SLOCK(X)	OK
T1	R1(Z)	SLOCK(Z)	OK
T2	R2(Y)	SLCOCK(Y)	OK
T3	R3(Y)	SLCOCK(Y)	OK
T1	W1(X)	XLOCK(X)	Non, T1 attend la fin de T3
T2	W2(Z)	XLOCK(Z)	Non, T2 attend la fin de T1
T3	W3(Y)	XLOCK(Y)	Non, T3 attend la fin de T2
T2	W2(Y)	XLOCK(Y)	T2 est en attente

- **Graphe d'attente**



- De graphe d'attente, on constate qu'on a le cycle {T3, T2, T1, T3} d'où on a un inter-blocage.

Pour lever l'inter blocage on choisit une transaction victime parmi {T1, T2, T3} on l'annule et on la relance à la fin. Les trois transactions n'ont pas fait de mise à jour. La transaction T3 est la victime (la plus récente).

Rollback(T3): UNLOCK(X), UNLOCK(Y) et réveiller T1			
Transaction	Action	Demande de verrous	Réponse
T1	W1(X)	XLOCK(X)	OK ; Fin de T1 ; Unlock(X), Unlock(Z) ; Réveiller T2
T2	W2(Z)	XLOCK(Z)	OK
T2	W2(Y)	XLOCK(Y)	OK ; Fin de T2 ; Unlock(Y) ; Unlock(Z) ;
<b>Relancer T3 dès son début :</b>			
T3	R3(X)	SLOCK(X)	OK
T3	R3(Y)	SLCOCK(Y)	OK
T3	W3(Y)	XLOCK(Y)	OK ; Fin de T3 ; Unlock(Y) ; Unlock(X) ;
L'exécution série équivalente: <b>T1 T2 T3</b> après le levé d'inter-blocage.			

3. Soient trois transactions T1, T2 et T3.

- a. Donner les différentes actions que le gestionnaire de transaction effectue à l'arrivée des événements Checkpoint 1 et Checkpoint 2 ainsi que les valeurs des granules A, B, C et D.

**a.1 CheckPoint1 :**

- Rendre effectives toutes les mises à jours de T1.
- Rendre effectives les mises à jours effectuées avant ce Checkpoint des transactions : T2.

**a.2 CheckPoint2 :**

- Rendre effectives toutes les mises à jours de T3.
- Rendre effectives les mises à jours effectuées entre Checkpoint 1 et Checkpoint 2 de la transaction T2.

**a.3 Etat des granules A, B, C et D**

**A→16 ; B→12 ; C→8 ; D→16 ;**

- b. Si on suppose l'arrivée d'une panne après la ligne 16. Donner l'état de chaque transaction juste avant l'arrivée de la panne

- Transactions validées: T1 et T3.
- Transactions actives: T2.