Interrogation SGBD (COURS)

Question 1

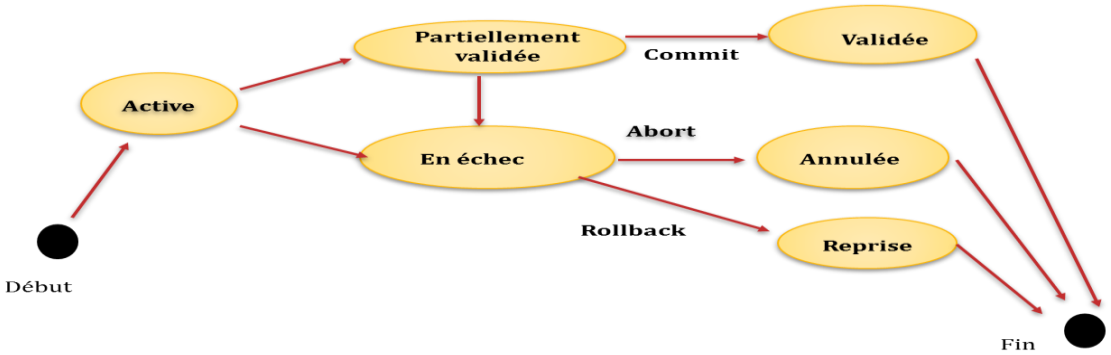
- Quelle est la différence entre un ordonnancement correct et un ordonnancement sérialisable?
- Donner le principe de la sérialisation par verrouillage? Quels sont ses avantages? Ses inconvénients?
- Quelle est la caractéristique des transactions qui sera violée lorsque le scheduler opte po ur une exécution entrelacée de transaction?

Question 2

- Décrire le cycle de vie d'une transaction?
- Quel est l'élément central dans la récupération de données? Comment il est alimenté?
- Pourquoi existe-t-il deux copies de cet élément, une en MS et une en MC?

Question 1

- La correction d'un ordonnancement est une condition nécessaire pour la sérialisabilité mais elle n'est pas suffisante. Donc, un ordonnancement sérialisable est correct, mais le contraire n'est pas forcément vrai. De plus, la sérialisabilité s'intéresse au résultat d'exécution ainsi que la construction (structure) de l'ordonnancement tandis que la correction s'intéresse uniquement au résultat obtenu.
- La sérialisation par verrouillage consiste à réserver l'accès à une donnée aux seules transactions qui l'ont demandée et dont le scheduler a donné un avis favorable. Une transaction ayant obtenu un avis favorable verrouille la donnée, par un verrou partagé si elle veut la lire ou un verrou exclusif si elle veut la modifier. Le scheduler ne pourra donner un avis favorable à une demande de verrou sur une donnée que dans les cas suivants :
  - La donnée est libre.
  - La donnée est verrouillée par un verrou partagé et que la transaction demandant l'accès veut lire cette donnée.
  - **Avantages**
    - Eviter les problèmes d'accès concurrents (mise à jour perdue, dépendance non valide, analyse incohérente)
    - Garantir l'exclusion mutuelle
    - La sérialisabilité peut être prouvée en utilisant le protocole V2P
  - **Inconvénients**
    - Verrous motels.
- Une exécution entrelacée de transactions affectera la caractéristique d'isolation des transactions.
- Cycle de vie d'une transaction :



- L’élément central dans la récupération des données est le journal de transaction. Il est alimenté par le gestionnaire de reprise à chaque exécution d’action dans une transaction. Il contient les informations suivantes : début des transactions, les actions exécutées, les images avant et après, les points de contrôle, les opérations d’annulation et les opérations commit.
- Le journal est manipulé au niveau de la MC. Une copie se trouve en MS, elle permet de matérialiser les modifications effectuées dans la version se trouvant en M. Il est important d’avoir la version en MS pour pouvoir le charger lorsqu’une panne survient. En effet, lorsqu’une panne arrive, la MC peut se vider (problème de courant, arrêt brusque de l’application ou du SGBD), dans ce cas le système charge la version en MS pour trouver les traces d’exécution des transactions.

Interrogation SGBD (TD)

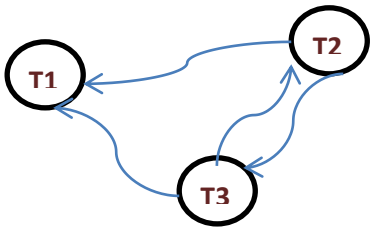
Exercice 1 :

Appliquer l'algorithme de verrouillage à deux phases sur les ordonnancements suivants

- **R1(A) W1(B) W3(A) R2(B) R3(B) W3(B) W1(B) R2(A)**
- **R1(A) R3(B) W1(B) W4(A) W5(A) R2(D) R3(D) W2(D) W5(D) R4(E) R5(E)**

- Vérifier dans chaque cas l'existence de deadlock, montrer comment le résoudre et montrer l'ordre d'exécution des transactions.
- Appliquer l'algorithme d'estampillage sur l'ordonnancement suivant :
- **R2(B) R1(A) W2(C) W2(B) R3(B) R3(C) R1(B) W1(A) W3(B) W1(B) W3(C)**

Action	Demande	Accord	Après fin T1
R1(A)	S1(A)	ok	-
W1(B)	X1(B)	ok	-
W3(A)	X3(A)	Non, T3 attend T1	Ok
R2(B)	S2(B)	Non, T2 attend T1	Ok
R3(B)	-		Ok
W3(B)	-		Non, T3 attend T2
W1(B)	X1(B)	Ok, Fin T1, U(A,B)	-
R2(A)	-		Non, T2 attend T3



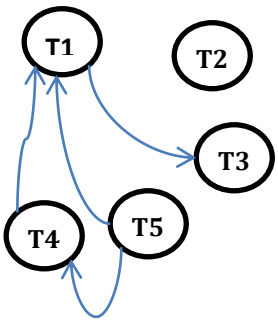
Il existe un deadlock. Annuler soit T2 ou T3. On choisit par exemple d'annuler T2 qui n'a pas fait de mise à jour. L'ordre d'exécution devient : T1, T3, T2.

Action	Demande	Accord	Après fin T3	Après fin T4
R1(A)	S1(A)	Ok		
R3(B)	S3(B)	Ok		
W1(B)	X1(B)	Non, T1 attend T3	Ok, Fin T1, U(B)	
W4(A)	X4(A)	Non, T4 attend T1	Ok	
W5(A)	X5(A)	Non, T5 attend T1	Non, T5 attend T4	Ok
R2(D)	S2(D)	Ok		
R3(D)	S3(D)	Ok, fin T3, U(B, D)		
W2(D)	X2(D)		Ok, Fin T2, U(D)	
W5(D)	X5(D)		-	ok
R4(E)	S4(E)		Ok, Fin T4, U(A, E)	
R5(E)	S5(E)			Ok, Fin T5, U(A,D,E)

Pas de deadlock.

Ordre d'exécution :

T3, T1, T2, T4, T5



Application de l'algorithme d'estampillage

	EL(A)	EE(A)	EL(B)	EE(B)	EL(C)	EE(C)
	0	0	0	0	0	0
R2(B)	0	0	**2**	0	0	0
R1(A)	**1**	0	2	0	0	0
W2(C)	1	0	2	0	0	**2**
W2(B)	1	0	2	**2**	0	2
R3(B)	1	0	**3**	2	0	2
R3(C)	1	0	3	2	**3**	2
R1(B)	1, **Annuler T1**	0	3	2	3	2
W1(A)	-	0	3	2	3	2
W3(B)	1	0	3	**3**	3	2
W1(B)	-	0	3	3	3	2
W3(C)	1	0	3	3	3	**3**
Après la relance de T1						
R4(A)	**4**	0	3	3	3	3
R4(B)	4	0	**4**	3	3	3
W4(A)	4	**4**	4	3	3	3
W4(B)	4	4	4	**4**	3	3

Interrogation SGBD (TD)

Exercice 2 :

Supposons le schéma de la BD « Laboratoire de Recherche » traitée en TP.

Laboratoire(CodeLab, NomLab, Chef_de_Lab*, Date-création, Siteweb).
Chercheur(NumCh, NomCh, CodeLab*, CodeUniversité*)
Université(CodeUniversité, NomUniversité)
Projet(CodePrj, Date-Début, Date-fin, Responsable, TypePrj*, Budget)
TypeProjet(CodeTypePrj, Libellé)
Paraticipe(NumCh*,CodePrj*, Charge_Horaire).

Un utilisateur (SGBD2011) veut exécuter deux requêtes SQL suivantes :

R1 :

```
SELECT NomLab
from SGBD.Laboratoire L, Chercheur C, SGBD.Participe R
where L.CodeLab=C.CodeLab and R.NumCh=C.NumCh and R.Charge_Horaire>100
```

R2 :

```
UPDATE SGBD.CHERCHEUR SET Code_Université = 'ESI' where Code_université ='INI'.
```

Pour chaque requête, répondez aux questions suivantes

- Donner toutes les vérifications effectuées par le SGBD pour répondre à cette requête (2+2).
- Pour chaque vérification, donner la requête que le SGBD pourra utiliser pour la réaliser (1+1).
- Dans le cas où la vérification échoue, proposer une solution que l'administrateur de la base de données pourra appliquer (1+1).
- Ecrire une procédure PlSQL qui permet à partir d'un numéro de chercheur, de calculer sa charge horaire totale (2).

Requête	R1	R2
Vérifications	1. Vérification lexico-syntaxique 2. L'existence des tables Laboratoire et Participe chez l'utilisateur SGBD 3. L'existence de la table Chercheur chez l'utilisateur SGBD2011 4. L'existence des attributs des trois tables 5. Le privilège SELECT sur Laboratoire et Participe pour l'utilisateur SGBD2011	1. Vérification lexico-syntaxique 2. L'existence de la table Chercheur chez l'utilisateur SGBD 3. L'existence de l'attribut Code_Université dans la table Chercheur 4. Le privilège UPDATE sur Chercheur pour l'utilisateur SGBD2011
Requêtes SGBD	1. Consulter le lexique et la grammaire SQL 2. SELECT Table_Name From ALL_USER where Owner='SGBD' 3. SELECT Table_Name From ALL_USER where Owner='SGBD2011' 4. SELECT Table_Name, Column_name FROM ALL_TAB_COLUMNS Where Owner in ('SGBD', 'SGBD2011') and Table_Name in('Laboratoire','Participe','Chercheur') Order By Table_Name 5. SELECT Privilege FROM ALL_TAB_PRIVS WHERE GRANTEE='SGBD2011' and Table_Name in ('Laboratoire', 'Participe') and Table_Schema='SGBD'.	1. Consulter le lexique et la grammaire SQL 2. SELECT Table_Name From ALL_USER where Owner='SGBD'. 3. SELECT Table_Name, Column_name FROM ALL_TAB_COLUMNS Where Owner ='SGBD' and Table_Name = 'Chercheur' 4. SELECT Privilege FROM ALL_TAB_PRIVS WHERE GRANTEE='SGBD2011' and Table_Name ='Chercheur' and Table_Schema='SGBD'.
Solution	1. Corriger les erreurs lexicales et syntaxiques 2. Vérifier si le nom des tables est bien saisi. Corriger les noms ou bien créer la table correspondante	1. Corriger les erreurs lexicales et syntaxiques 2. Vérifier si le nom de la table est bien saisi. Corriger le nom ou bien créer la table

	3. Même chose 4. Vérifier si le nom des attributs est bien saisi. Corriger les noms ou bien ajouter les attributs. 5. Attribuer le privilège (Grant SELECT ON SGBD.Laboratoire,SGBD.Participe TO SGBD2011)	3. Vérifier si le nom des attributs est bien saisi. Corriger les noms ou bien ajouter les attributs. 4. Attribuer le privilège (Grant UPDATE ON SGBD.Chercheur TO SGBD2011)

Programme plSQL

```
CREATE OR REPLACE FUNCTION CalculeCharge(number NCH)
cursor cr is select  Charge_Horaire from CHERCHEUR where NumCh=NCH;
 c_rec cr%rowtype;
ChargeH binary_integer;
SansCharge EXCEPTION;
BEGIN
   ChargeH := 0;
   for c_rec in cr loop
            ChargeH := ChargeH + c_rec.Charge_Horaire;
            exit when cr%notfound;
   end loop;
if(ChargeH =0) then RAISE SansCharge;
else
RETURN ChargeH ;
end if;
EXCEPTION     WHEN SansCharge THEN
    dbms_output.put_line ('Attention! Le Chercheur N°  ' ||NCHi||' est sans charge');
END
```