USTHB
Faculté d'Informatique 2025/2026

Examen Administration des SGBD

M1 : IL-SII-RSD

Enoncé :

On s'intéresse à la gestion des services douaniers. Ci-après le schéma simplifié de la base de données :

Brigade (**Num_Brig**, Wilaya, NB_Douanier, ***MatD_Chef****)

Douanier (**MatD**, NomD, PrenD, Date_nais, AdrD, Ntel, Grade, Salaire, Fonction, ***Num_Brig****)

Intervention (**Num_interv**, Date_interv, Lieu_inter, ***Num_Brig****)

Participer (**Num_interv*, MatD***)

Produit (**CodeP**, NomP, Qte_maxi_autorisé, Prix_unitaire)

Saisie (**CodeP*, Num_interv***, Qte_saisie)

Sachant que :

- Les clés primaires sont soulignées et * désigne une clé étrangère.
- Le garde d'un Douanier peut être : «AGENT», «COMMANDANT » ou «SERGENT».
- «MatD_chef » et «MatD» sont du même domaine.
- Le salaire d'un douanier ≥ 50 000 DA.

Exercice 1 : Fonctions générales de SGBD et Optimisation (11 points)

1. Décrire l'architecture interne d'un SGBD en précisant :
   - le niveau externe,
   - le niveau conceptuel,
   - le niveau interne.

Illustrer chaque niveau par un exemple tiré de la base des services douaniers.

Réponse : (1.5 pts)

a. Niveau externe

- Correspond aux **vues utilisateur**.
- Chaque utilisateur a une vision partielle de la base.

Exemple :
Vue donnant au chef de brigade uniquement les informations sur les douaniers de sa brigade.

b. Niveau conceptuel

- Décrit la **structure logique globale** de la base.
- Indépendant des utilisateurs et du stockage.

Exemple :
Schéma conceptuel de l'énoncé

c. Niveau interne

- Décrit l'**organisation physique** des données (fichiers, blocs, index).

Exemple :

- Table Intervention stockée sur 10 blocs disque
- Index sur Num_interv

2. Supposant que la table Douanier est créée sans contraintes d'intégrité :
   a. écrire les commandes SQL permettant d'ajouter toutes les contraintes nécessaires.

Réponse : (2 pt)

```
ALTER TABLE Douanier ADD CONSTRAINT PK_Douanier PRIMARY KEY (MatD);

ALTER TABLE Douanier ADD CONSTRAINT FK_Douanier_Brigade FOREIGN KEY (Num_Brig) REFERENCES Brigade(Num_Brig);

ALTER TABLE Douanier ADD CONSTRAINT CK_Grade CHECK (Grade IN ('AGENT', 'COMMANDANT', 'SERGENT'));

ALTER TABLE Douanier ADD CONSTRAINT CK_Salaire CHECK (Salaire >= 50000);
```

   b. Utiliser le dictionnaire de donner pour afficher les contraintes crées.

Réponse : (0.5pt)

```
SELECT constraint_name, constraint_type
FROM user_constraints
WHERE table_name = 'DOUANIER';
```

3. Exprimer, à l'aide de contraintes, la règle suivante :
   « Le salaire d'un douanier ne peut pas être supérieur au salaire du chef de sa brigade. »

Réponse : (1pt)

```
CREATE OR REPLACE TRIGGER TR_Salaire
BEFORE INSERT OR UPDATE ON Douanier
FOR EACH ROW
DECLARE
 v_salaire_chef NUMBER;
BEGIN
 SELECT Salaire INTO v_salaire_chef
 FROM Douanier
 WHERE MatD = (
  SELECT MatD_Chef
  FROM Brigade
  WHERE Num_Brig = :NEW.Num_Brig
 );

 IF :NEW.Salaire > v_salaire_chef THEN
  RAISE_APPLICATION_ERROR(-20001,'Salaire invalide');
 END IF;
END;
/
```

4. On souhaite créer une vue donnant, pour chaque brigade, le nombre total d'interventions effectuées.
   a. Donner la commande SQL de création de la vue.

Réponse : (0.5pt)

```
CREATE VIEW V_NB_INTERV AS
SELECT Num_Brig, COUNT(*) AS Nb_Interv
FROM Intervention
GROUP BY Num_Brig;
```

b. Indiquer les répercussions de cette création sur le catalogue système.

Réponse : (0.5 pt)

Ajout dans `USER_VIEWS, DBA_VIEWS, ALL_VIEWS`
`Si on considère les catalogues relation et vue`
`Ajout d'une ligne dans le catalogue vue`
`Cardinalite de la relation vue dans le catalogue relation +1`

c. On souhaite mettre en place un contrôle d'accès strict pour les chefs de brigade.

Un chef de brigade doit pouvoir consulter uniquement le nombre total d'interventions par brigade, sans avoir accès aux détails des interventions ni à la table *Intervention*.

1. Créer un rôle CHEF_BRIGADE.
2. Retirer tout droit d'accès direct à la table *Intervention* pour ce rôle.
3. Autoriser ce rôle à consulter uniquement le nombre total d'interventions par brigade.

Donner les commandes SQL nécessaires.

Réponse : 0.75 pt

```
CREATE ROLE CHEF_BRIGADE;
REVOKE SELECT ON Intervention FROM CHEF_BRIGADE;
GRANT SELECT ON V_NB_INTERV TO CHEF_BRIGADE;
```

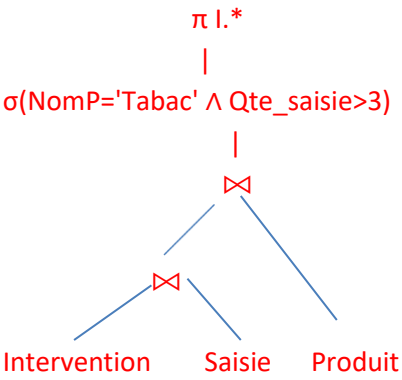
6. On cherche les interventions où du tabac a été saisi avec une quantité supérieure à 3 tonnes.
   a. Donner la requête SQL correspondante.

Réponse : 0.5 pt

```
SELECT I.*
FROM Intervention I, Saisie S, Produit P
WHERE I.Num_interv = S.Num_interv
  AND S.CodeP = P.CodeP
  AND P.NomP = 'Tabac'
  AND S.Qte_saisie > 3;
```

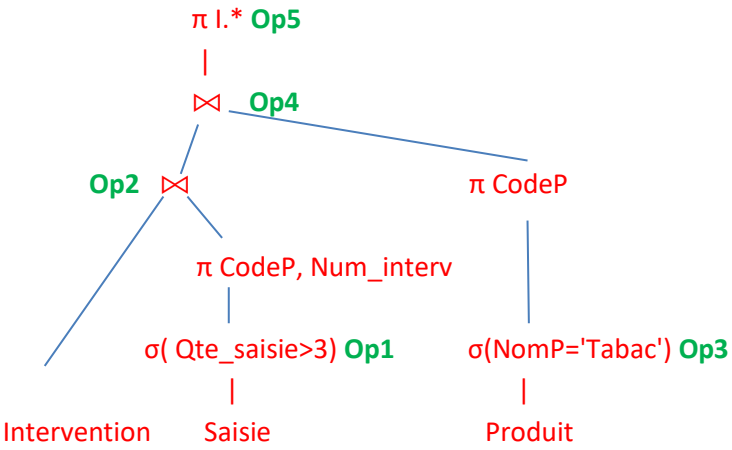
b. Construire l'arbre algébrique non optimisé associé.

Réponse : 0.5pt



c. Générer l'arbre algébrique optimisé en appliquant les règles de transformation.

Réponse : 0.75 pt



d. Calculer le coût d'exécution correspondant au plan optimisé de cette requête.

Statistique sur les données :

Card(Intervention)= 10000 tuples	Nombre de bloc pour Intervention : 1000
Card(Produit)=100 tuples	Nombre de bloc pour Produit : 10
Card(Saisie)= 1000 tuples	Nombre de bloc pour Saisie : 10
Val(Produit, NomP)=100	
Max(Qte_saisie)=10	Min(Qte_saisie)=1
FS(A>val)= Max(A)-val/Max(A)-Min(A)	

Réponse : 2.5 pts (0,5 pour chaque opération)

(J'accepte différents calcul selon différents arbres)

Saisie : S, Produit : P, Intervention : I

Opération	Coût d'exe	Cardinalité
Op1	|S|= 10	||S||* Max(S)-3/Max(S)-Min(S) = 1000* (10- 3)/(10-1) =778 tuples
Op2	|I|=1000 Le résultat de Op1 est en MC	||I||*||Op1||/Max(val(I,NumInter), val(S,numInter)) =||Op1||= 778 tuples
Op3	|P|=10	||P||*1/val(P,NomP)= 100/100=1
Op4	0 (en MC)	||Op2||*||Op3||/ Max(val(S,codeP), val(P,codeP)) 778*1/100=8 tuples
Op5	0 (en MC)	||Op4||= 8 tuples
Total	1020 E/S	8 tuples

Exercice 2 : Gestions des accès concurrents (09 points)

1. Considérez l'ordonnancement **O1** suivant :

R1(Z) R3(X) W3(Z) W4(X) R1(X) W2(X) W2(V) W5(V) W5(Y) R1(Y).

   a. Dites si **O1** est sérialisable ou non, en construisant le graphe de précédence.

Réponse : 2 pt

Les conflits:

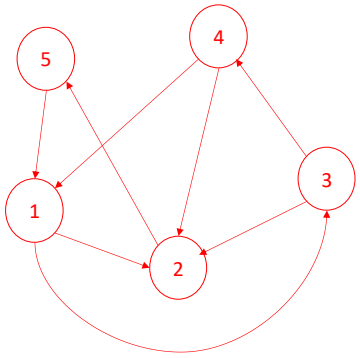
X: R3(X)-W4(X), R3(X)-W2(X)

W4(X)-R1(X), W4(X)-W2(X), R1(X)-W2(X)

Y: W5(Y)-R1(Y)

Z: R1(Z)-W3(Z)

V: W2(V)-W5(V)



Présence de cycles : (2-5-1-3-2), (3-4-1-3) (1-2-5-1) O1 n'est pas sérialisable.

Si **O1** est sérialisable, décrivez tous les ordonnancements en série qui sont équivalents à **O1**. Sinon, déterminez quel est le nombre minimal de transactions qui doivent être supprimées de **O1** pour que l'ordonnancement résultant soit sérialisable. Précisez ces transactions.

Réponse : 0,5 pt

On remarque que les nœuds 1 est commun aux 3 cycles, il suffit donc de supprimer le noeud pour éliminer les cycles.
Donc le nombre minimale de transactions à supprimer pour obtenir la sérialisabilité est 1 : T1.

2. Considérez l'ordonnancement **O2** suivant :

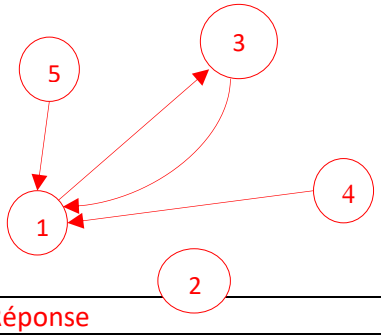
R4(Y) W1(Z) R2(Y) W3(X) W1(Y) R1(X) R3(Z) C3 W5(Z) C2 W5(Y) C5 W4(Z) C1 R4(X) C4

   a. Donnez le scenario d'exécution de cet ordonnancement dans le cas du verrouillage à deux phases. Construisez le graphe d'attente de cet ordonnancement. Existe-t-il un deadlock ? Si oui, proposez une solution à ce problème.

Réponse : 3 pt

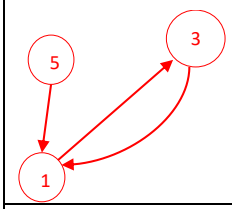
Action	Demande de verrou	Réponse
R4(Y)	Slock(Y)	Ok
W1(Z)	Xlock(Z)	Ok
R2(Y)	Slock(Y)	Ok
W3(X)	Xlock(X)	Ok
W1(Y)	Xlock(Y)	Non. T1 attend T2 et T4
R1(X)	-	T1 en attente
R3(Z)	Slock(Z)	Non, T3 attend T1
C3	-	T3 en attente
W5(Z)	Xlock(Z)	Non, T5 attend T1
C2	-	Fin T2. Unlock(Y)
W5(Y)	-	T5 en attente
C5	-	T5 en attente
W4(Z)	Xlock(Z)	Non, T4 attend T1
C1	-	T1 en attente
R4(X)	-	T4 en attente
C4	-	T4 en attente

Le graphe d'attente contient un cycle, il existe donc un interblocage.
Solution : on annule T1 ou T4 et on la refait à la fin.



Si T4 est annulé, T1 est réveillé.

Action	Demande de verrou	Réponse
Réveiller T1		
W1(Y)	Xlock(Y)	ok
R1(X)	Slock(X)	Non, T1 attend T3
Graphe d'attente		
Réveiller T1		
R1(X)	Slock(X)	Non, T1 attend T3
C1	-	Fin T1. Unlock(X,Y,Z). Réveiller T5
W5(Z)	Xlock(Z)	Ok.
W5(Y)	Xlock(Y)	ok
C5	-	Fin de T5. Unlock(Z,Y)
Refaire T4		
R4(Y)	Slock(Y)	ok
W4(Z)	Xlock(Z)	ok
R4(X)	Slock(X)	ok
C4	-	Fin T4. Unlock (Y,Z,X)
Refaire T3 W3(X) R3(Z) C3		



Le graphe d'attente contient un cycle, il existe donc un interblocage.
Solution : on annule T1 ou T3et on la refait à la fin.

b. Si la stratégie wait-die est utilisée pour gérer les demandes de verrouillage. Quelle est l'exécution finale obtenue ?

Si on affecte les estampilles des transactions selon l'ordre de démarrage des transaction, nous avons : E(T4)<E(T1)<E(T2)<E(T3)<E(T5)

Réponse : 1,5 pt

Action	Demande de verrou	Réponse
R4(Y)	Slock(Y)	Ok
W1(Z)	Xlock(Z)	Ok
R2(Y)	Slock(Y)	Ok
W3(X)	Xlock(X)	Ok
W1(Y)	Xlock(Y)	Non, T1 est annulé (plus récente que T4). Unlock(Z). Refaire T1 à la fin.
R1(X)	-	T1 annulé
R3(Z)	Slock(Z)	Ok
C3	-	Fin T3. Unlock(X,Z)
W5(Z)	Xlock(Z)	Ok
C2	-	Fin T2. Unlock(Y)
W5(Y)	Xlock(Y)	Non, T5 est annulée (plus récente que T4). Unlock(Z). Refaire T5 à la fin.
C5	-	Fin T5. Unlock(Z, Y)
W4(Z)	Xlock(Z)	ok
R4(X)	Slock(X)	ok
C4	-	Fin T4
Refaire T1		
Refaire T5		

@ zakia : pour moi la condition est exécutée quel que soit la demande lecture ou écriture voici mon déroulement pour la wait and die (plus ancienne attend plus récente s'annule)

Réponse :

Action	Demande de verrou	Réponse
R4(Y)	Slock(Y)	Ok
W1(Z)	Xlock(Z)	Ok
R2(Y)	Slock(Y)	Non, T2 est annulée (plus récente que T4). Refaire T2 à la fin.
W3(X)	Xlock(X)	Ok
W1(Y)	Xlock(Y)	Non, T1 est annulée (plus récente que T4). Unlock(Z). Refaire T1 à la fin.
R1(X)	-	T1 annulée
R3(Z)	Slock(Z)	Ok
C3	-	Fin T3. Unlock(X,Z)
W5(Z)	Xlock(Z)	Ok
C2	-	T2 annulée
W5(Y)	Xlock(Y)	Non, T5 est annulée (plus récente que T4). Unlock(Z). Refaire T5 à la fin.
C5	-	T5 annulée
W4(Z)	Xlock(Z)	Ok
R4(X)	Slock(X)	Ok
C4	-	Fin T4
Refaire T2		
Refaire T1		
Refaire T5		

3. Dans le SGBD **M**, chaque transaction $\mathbf{T_i}$ possède un identifiant **id(T_i)** qui est un nombre entier. Le mécanisme de contrôle de concurrence de **M** est tel que, dans un ordonnancement **O**, une transaction $\mathbf{T_i}$ est autorisée à **écrire après** une autre transaction $\mathbf{T_j}$ (c'est-à-dire que le mécanisme ne bloque pas l'ordonnancement s'il contient une action d'écriture de $\mathbf{T_j}$ sur un élément **X** telle que l'action suivante sur **X** dans **O** est une écriture de T_i sur **X**) **si et seulement si id(T_i) > id(T_j)**. De même, $\mathbf{T_i}$ est autorisée à **lire après** une transaction $\mathbf{T_h}$ (c'est-à-dire que le mécanisme ne bloque pas l'ordonnancement s'il contient une action d'écriture de $\mathbf{T_h}$ sur un élément **X** dans **O** telle que l'action suivante sur **X** est une action de lecture de $\mathbf{T_i}$ sur **X**) **si et seulement si id(T_i) > id(T_h)**.

a. **Prouvez ou réfutez** que tout ordonnancement accepté par le mécanisme de contrôle de concurrence de **M** est sérialisable par graphe de précédence.

Réponse : 2pt

Pour prouver qu'une affirmation est fausse, il suffit de trouver un seul cas (un ordonnancement) qui respecte les règles du mécanisme M mais qui n'est pas sérialisable par graphe de précédence.

1. Identification de la faille

Le mécanisme M impose des conditions sur :

- Les conflits **Écriture-Écriture (W-W)** : T_j écrit, puis T_i écrit.
- Les conflits **Écriture-Lecture (W-R)** : T_h écrit, puis T_i lit.

Cependant, il n'impose **aucune condition** sur les conflits **Lecture-Écriture (R-W)** (quand une transaction lit une donnée avant qu'une autre ne la modifie).

2. Construction du contre-exemple

Soient deux transactions T_1 et T_2 avec les identifiants :

- $id(T_1) = 10$
- $id(T_2) = 20$

Considérons l'ordonnancement **O** suivant portant sur deux objets X et Y :

O : W1(x) R2(X) R2(Y) W1(Y)

3. Vérification des règles de M

Vérifions si cet ordonnancement est accepté par le mécanisme :

1. **Conflit sur X : W1(X)-R2(X) :** C'est un conflit de type Écriture-Lecture. Le mécanisme l'autorise si $id(T_2) > id(T_1)$. Comme 20 > 10, cette étape est **acceptée**.
2. **Conflit sur Y : R2(Y)-W1(Y) :** C'est un conflit de type Lecture-Écriture. Le mécanisme M ne prévoit aucune restriction pour ce cas. Cette étape est donc **acceptée** par défaut.

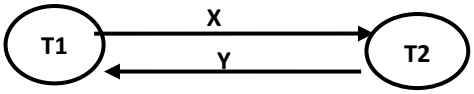
L'ordonnancement O est donc entièrement **accepté par le mécanisme de concurrence M**.

4. Analyse de la sérialisabilité par conflit

Un ordonnancement est sérialisable par conflit s'il n'y a pas de cycle dans son graphe de précédence. Analysons les conflits dans O :

Conflit sur X : W1(X)-R2(X).

Conflit sur Y : R2(Y)-W1(Y).



Conclusion

Puisque le graphe de précédence contient un cycle, l'ordonnancement O n'est **pas sérialisable par conflit**. Comme le mécanisme M accepte cet ordonnancement, nous avons prouvé par l'absurde que **tous les ordonnancements acceptés par M ne sont pas nécessairement sérialisables par conflit**.

L'affirmation est donc **réfutée**.