USTHB
Faculté d'Informatique 2024/2025

Examen Administration des SGBD

M1 : IL-SII-MIV

Enoncé :

Une entreprise utilise une base de données relationnelle décrite par le schéma relationnel ci-dessous pour gérer les drones et missions, avec différents types d'utilisateurs : administrateurs, opérateurs de drones, superviseurs et contrôleur de qualité.

Modele (**modeId**, modelnom)

Drone (**numd**, *modeld*, dateaqcui, statut);

Mission (**codem**, *numd*, *typemid*, *zid*, datedeb, datefin);

Utilisateurs (**idp**, nom, prenom, email, role) ;

TypeMission (**typemid**, type) ;

Zone (**zid**, region, coordonnees) ;

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

- Statut : cette colonne indique si le drone est : Disponible, En Maintenance ou En Mission.
- La colonne Type de mission prend les valeurs : Agriculture, Sécurité ou Environnement ;
- Rôle peut être : administrateurs, opérateurs de drones, superviseurs ou contrôleur de qualité
- Région : Est, Ouest, Sud ou Nord.

Nous disposons des informations suivantes :

Modele	Card(Modele)=1000
Drone	Card(Drone)=100000
Mission	Card(Mission)=10000000
Utilisateur	Card(Utilisateur)=100
TypeMission	Card(TypeMission)= 3
Zone	Card(Zone)= 4000
On suppose que chaque attribut est codé sur 50 octets. Que la base de données contient les missions des drones datées entre le 01-01-2020 et 31-12-2024.	

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

1. Le DBA exécute les requêtes suivantes :

```
CREATE ROLE ROLE_ADMIN;
CREATE ROLE ROLE_OPERATEUR;
CREATE ROLE ROLE_SUPERVISEUR;
CREATE ROLE ROLE_CONTROLEUR_QUALITE;
GRANT ALL PRIVILEGES ON Drone TO ROLE_ADMIN;
GRANT ALL PRIVILEGES ON Mission TO ROLE_ADMIN;
GRANT ALL PRIVILEGES ON Utilisateur TO ROLE_ADMIN;
GRANT SELECT ON Drone TO ROLE_OPERATEUR;
GRANT INSERT, UPDATE ON Mission TO ROLE_OPERATEUR;
GRANT SELECT ON Drone TO ROLE_SUPERVISEUR;
GRANT SELECT ON Mission TO ROLE_SUPERVISEUR;
GRANT SELECT ON Drone TO ROLE_CONTROLEUR_QUALITE;
GRANT UPDATE (status) ON Drone TO ROLE_CONTROLEUR_QUALITE;
```

```
CREATE PROFILE PROFIL_OPERATEUR
LIMIT
  SESSIONS_PER_USER 2
  CONNECT_TIME 120
  PRIVATE_SGA 50M;
CREATE USER admin IDENTIFIED BY admin_pass;
GRANT ROLE_ADMIN TO admin;
CREATE USER operateur IDENTIFIED BY psw;
GRANT ROLE_OPERATEUR TO operateur;
ALTER USER operateur PROFILE PROFIL_OPERATEUR;
CREATE USER superviseur IDENTIFIED BY psw;
GRANT ROLE_SUPERVISEUR TO superviseur;
CREATE USER controleur IDENTIFIED BY psw;
GRANT ROLE_CONTROLEUR_QUALITE TO controleur;
```

   a. Quels privilèges l'utilisateur opérateur possède-t-il ? **(0.75pt)**

      Quelles sont les limitations imposées par son profil ? **(0.75pt)**

L'utilisateur opérateur peut lire la table Drone, ajouter et modifier des lignes dans la table Mission.

Cet utilisateur peut ouvrir deux sessions au maximum ; chaque session dure deux heures. Il a un quota de 50M dans l'espace mémoire SGA.

b. Un superviseur essaie de modifier une mission ? Quelle serai la réponse du SGBD. Justifiez. **(0.5pt)**

Le SGBD refuse cette modification car le superviseur possède que le privilège de lecture sur la table Mission.

c. L'utilisateur contrôleur exécute la requête suivante grant UPDATE (status) ON Drone TO superviseur ; Quelle serai la réponse du SGBD. Justifiez. **(0.5pt)**

Le SGBD refuse d'exécuter cette requête car le superviseur a reçu le privilège sans le droit de retransmission.

2. Donnez le script SQL pour créer la table **mission** avec toutes les contraintes d'intégrité sachant que les autres tables sont déjà créées. **(0.75pt)**

```
CREATE TABLE Mission ( codem varchar(10), numd INT NOT NULL, typemid INT NOT NULL, zid INT NOT NULL, datedeb DATE NOT NULL, datefin DATE,
CONSTRAINT PK_MISSION PRIMARY KEY (codem),
CONSTRAINT FK_DRONE FOREIGN KEY (numd) REFERENCES Drone(numd),
CONSTRAINT FK_TYPEMISSION FOREIGN KEY (typemid) REFERENCES TypeMission(typemid),
CONSTRAINT FK_ZONE FOREIGN KEY (zid) REFERENCES Zone(zid),
CONSTRAINT CK_DATE CHECK(datedeb<datefin) );
```

3. L'administrateur souhaite suivre le nombre total de missions affectées à chaque drone.

a. Ecrivez une fonction qui calcule le nombre de missions affectées à un drone. **(1pt)**

```
CREATE FUNCTION CountMissions(numdrone%Drone.numd%type) RETURNS INT is
DECLARE mission_count INT;
BEGIN
 SELECT  COUNT(*) into mission_count FROM Mission WHERE numd = numdrone;
RETURN mission_count;
END ;
```

b. Comment garantir que le nombre de missions affectées à un drone ne dépasse pas 100 missions ? **(1pt)**

```
CREATE TRIGGER LimitMissions
ON Mission
BEFORE INSERT OR UPDATE of (numd)
FOR EACH ROW
DECLARE
err EXCEPTION;
BEGIN
if (CountMissions(:new.numd)>100) then RAISE err;
end if;
EXCEPTION
```

```
WHEN err THEN raise_application_error(-20102,'le nombre de missions affectés à un drone ne doit pas dépasser 100');
WHEN OTHERS then  DBMS_OUTPUT.PUT_LINE('error : '||sqlcode||' '||sqlerrm);
END;
/
```

4. L'administrateur a constaté que les requêtes les plus fréquentes sur la table mission utilisent les colonnes codem et datedeb.
   a. Écrivez la commande SQL pour créer un index B-Arbre sur la colonne datedeb. **(0.5pt)**

```
CREATE INDEX DATEDEB_IX ON Mission(datedeb);
```

   b. Expliquez pourquoi un index B-Arbre est mieux adapté qu'un index bitmap pour une colonne comme datedeb. **(0.5pt)**

un index B-Arbre est mieux adapté qu'un index bitmap dans ce cas, car :
- L'attribut **datedeb** est une colonne avec une large gamme de valeurs distinctes.
- Les index bitmap sont plus efficaces pour des colonnes ayant un faible nombre de valeurs distinctes.

   c. Si la colonne datedeb subit des mises à jour fréquentes, comment cela peut-il affecter les performances de l'index B-Arbre ? **(0.5pt)**

Les mises à jour fréquentes de la colonne **datedeb** peuvent ralentir les performances en raison de la **réorganisation régulière** de l'arbre.

   d. Pourquoi l'index B-Arbre est particulièrement utile pour les requêtes utilisant ORDER BY ? **(0.5pt)**

Il maintient les valeurs indexées dans un ordre trié, ce qui évite des opérations de tri coûteuses.

5. En exécutant la requête ci-dessous ; L'administrateur a constaté qu'elle est lente.

```
SELECT distinct d.numd, d. modeld
FROM Drone d, Mission m, TypeMission t
WHERE d.numd = m.numd
AND  m. typem=t. typemid
AND m.datedeb > '01-01-2024'
AND t.type = 'Agriculture'
ORDER BY d.numd, d. modeld DESC;
```

   a. Que calcule cette requête ? **(0.5pt)**

La requête affiche une liste triée ordre décroissant des drones et leurs modèles associés à des missions de type "Agriculture" datées après le 01/01/2024.

   b. Quels sont les facteurs qui peuvent ralentir cette requête ? **(0.75pt)**

Facteurs de ralentissement :
- Absence d'index sur les colonnes utilisées dans les conditions des restrictions.
- Jointures entre plusieurs tables et l'ordre de jointure n'est pas optimal ; il faut commencer avec les table de petite cardinalité ?
- Opération DISTINCT qui nécessite un tri.

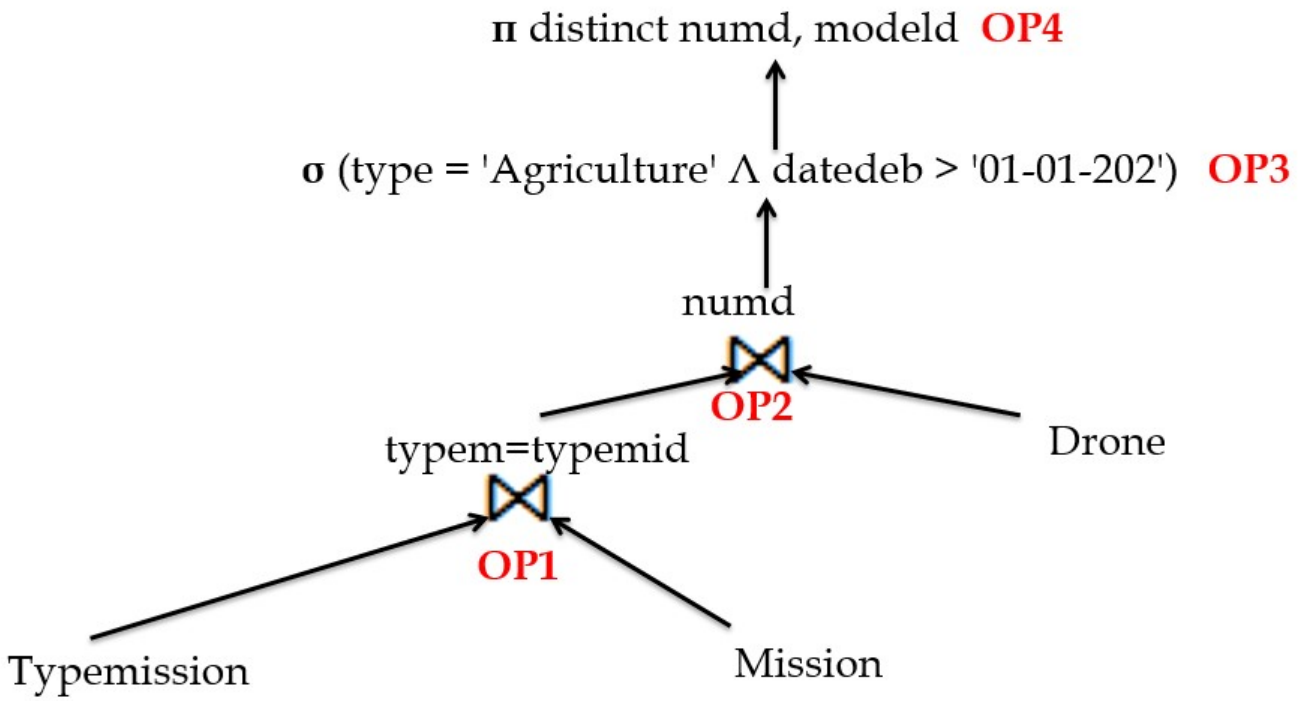
c. Indiquez les index à créer pour améliorer le temps de réponse de cette requête. **(1pt)**

Les index recommandés

- Un index secondaires pour accélérer les sélections sur l'attribut **datedeb** de la table mission.
- Des index primaire sur les attributs de jointure pour accélérer les jointures par exemple numd de la table Drone.

Donnez un plan logique de cette requête **(0.5pt)**

et calculer la taille de résultat. **(2pt)**



- Card(OP1) ; jointure sur clé étrangère card(OP1)= card(mission)= 10000000 ;
- Card(OP2) ; jointure sur clé étrangère card(OP2)= card(mission)= 10000000 ;
- Card(OP3)=card(OP2)* (1 /3)*[(31-12-2024-01-01-2024)/(31-12-2024-01-01-2020)]
  =10000000 *(1/3)*(365/1826)= 666 301
- Card(OP4)= MIN [Val (OP3, numd)*Val(OP3, modeld), 1/2*Card(OP3)]*(size(numd)+size(modeld))
  = MIN (1000*100000, (1/2)* 666301)*100
  =(1/2)* 666301*100=33 315 050 octets

NB: **-** Pour A>c, le Facteur de Sélectivité est défini comme suit (**Max(A)-c) / (Max(A) - Min(A)**).

- Pour éliminer les doublons d'une relation R (a1, …, an) l'estimation de taille du résultat est donnée par la formule suivante **MIN [Val (R, a1)*…*Val(R, an), 1/2*Card(R)]**.

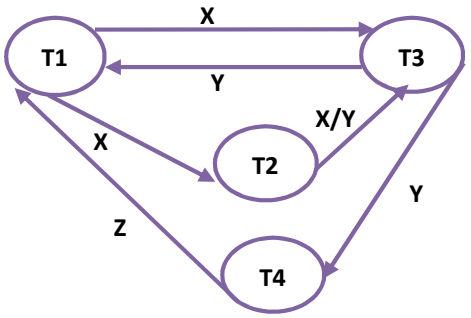
Exercice 2 : Gestions des accès concurrents (08 points)

1. Considérons l'ordonnancement suivant R1(X) W4(Z) W2(X) R2(Y) W3(Y) W3(X) R1(Y) R4(Y) W1(Z)

   a. Vérifiez si cet ordonnancement est sérialisable en identifiant les conflits et en construisant le graphe de précédence. **(1.5pt)**

Les conflits :

- Sur X : R1(X)-W2(X) ; R1(X)-W3(X) ; W2(X)-W3(X)
- Sur Y : R2(Y)-W3(Y) ; W3(Y)-R1(Y) ; W3(Y)-R4(Y) ;
- Sur Z : W4(Z)-W1(Z) ;



Le graphe de précédence contient le cycle {T1, T3, T1} d'où cet ordonnancement n'est pas sérialisable.

   b. L'algorithme d'estampillage avec deux estampilles accepte cet ordonnancement sans rejet ? **(1.5pt)**

Transaction	Action	EL(X)	EE(X)	EL(Y)	EE(Y)	EL(Z)	EE(Z)
		0	0	0	0	0	0
T1	R1(X)	1	0	0	0	0	0
T4	W4(Z)	1	0	0	0	0	4
T2	W2(X)	1	2	0	0	0	4
T2	R2(Y)	1	2	2	0	0	4
T3	W3(Y)	1	2	2	3	0	4
T3	W3(X)	1	3	2	3	0	4
T1	R1(Y)	EL(Y)>E(T1) ET EE(Y)>E(T1) ; d'où R1(Y) est rejetée et T1 est annulée ; L'algorithme s'exécute avec rejet.					

2. Considérons les transactions suivantes :

T1 : R1(A) W1(C) ;	**T2 : R2(B) W2(A) ;**	**T3 : R3(C) W3(D) ;**	**T4 : W4(D) R4(B) ;**

Répondez aux questions suivantes, en fournissant une justification détaillée pour chaque réponse.

   a. Combien d'ordonnancements non sérialisables peut-on obtenir avec T1, T2, T3 et T4 ? **(1pt)**

Un ordonnancement est sérialisable si son graphe de précédence n'a pas de cycle.

T1→T2 (conflit sur A)

T1→T3 (conflit sur C)

T3→T4 (conflit sur D)

Aucun cycle n'est présent dans ce graphe.

Tous les ordonnancements possibles sur T_1, T_2, T_3 et T_4 sont sérialisables.

b. Montrez qu'en utilisant le mécanisme d'accès par verrouillage à deux phases, aucun ordonnancement de T1, T2, T3 et T4 ne peut engendrer un blocage mutuel ? **(1pt)**

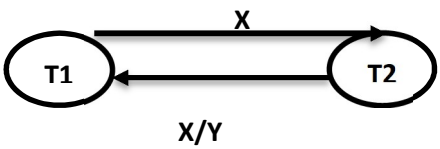
Les quatre transactions contiennent chacune deux actions et utilise deux granules différents.
Le conflit entre T1 et T2 génère une attente momentanée car la deuxième action termine la transaction et débloque celle qui en attente (T1 ou T2) en libérant le verrou sur A.
De même le conflit entre T1 et T3 se traduit pour une attente qui sera débloqué lorsque la transaction active (T1 ou T3) se termine et libère le verrou sur C.
Le conflit entre T3 et T4 aussi est géré par une attente qui se termine une fois la transaction (T3 ou T4) arrive à sa fin et libère le verrou sur D.
D'où aucun ordonnancement de T1, T2, T3 et T4 ne peut engendrer un blocage mutuel.

3. Considérons un contexte avec des verrous partagés et exclusifs. Une transaction **T** est dite **prudente** si toutes ses demandes de verrouillage apparaissent avant toute autre action de **T (par exemple T1 : SLOCK1(X) XLOCK1(Y) XLOCK1(Z) R1(X) W1(Y) W1(Z)).** Un ordonnancement est dit **prudent** si toutes ses transactions sont prudentes. Un ordonnanceur est dit **conservateur** s'il se comporte comme un ordonnanceur V2P avec la seule restriction supplémentaire qu'il n'accepte pas les ordonnancements qui ne sont pas prudents.

Prouvez ou réfutez les affirmations suivantes.

a. Tout ordonnancement prudent est sérialisable. **(1pt)**

Soit l'ordonnancement O1 prudent suivant : S1(X) X1(Y) R1(X) X2(Y) X2(X) W2(X) W2(Y) W1(Y)
Un ordonnancement est sérialisable si son graphe de précédence n'a pas de cycles.
L'ordonnancement O1 est prudent mais le graphe de précédence contient un cycle



Tous les ordonnancements prudents ne sont pas sérialisables.

b. Tout ordonnanceur conservateur peut éviter de gérer les interblocages. **(1pt)**

Dans un ordonnancement prudent, une transaction T demande tous ses verrous avant d'exécuter toute action. Si T doit attendre un verrou détenu par une autre transaction, T n'a encore exécuté aucune action. Cela signifie que T n'a pas d'arêtes entrantes dans le graphe d'attente.
Une fois T en cours d'exécution, elle a déjà acquis tous ses verrous. Elle ne sera donc jamais bloquée en attente d'un autre verrou. Cela signifie que T n'a pas d'arêtes sortantes dans le graphe d'attente.
Ainsi, le graphe d'attente est acyclique, garantissant l'absence de blocage.
Oui, Un ordonnanceur conservateur peut éviter de gérer les blocages.

c. Comparez les deux types d'ordonnanceurs V2P et conservateur ? En termes Sérialisabilité, interblocages et concurrence. **(1pt)**

Sérialisabilité et interblocages : Les deux types garantissent la sérialisabilité. Cependant, les ordonnanceurs conservateur évitent complètement les interblocages en refusant des ordonnancements qui pourraient les causer.

Concurrence : Les ordonnanceurs V2P offrent une meilleure concurrence, permettant une acquisition progressive des verrous, ce qui est utile dans des environnements dynamiques. En revanche, les ordonnanceurs conservateurs imposent des restrictions strictes, limitant les opportunités de parallélisme.

Bon courage