Systèmes de gestion de bases de données (NFP107)
Tout document écrit est autorisé
Examen de rattrapage de 2 heures et 30 minutes

1 Algèbre et SQL (10 points)

Soit le schéma relationnel suivant, utilisé pour la base d'un fleuriste :

Variété (Num_Variété, Nom_Variété, Couleur, Période_fraîcheur)
Stock (Num_Variété, Date_arrivée, Nombre_fleurs)
Personne (Num_Personne, Nom, Prénom, Adresse)
Commande (Num_Personne, Num_Variété, Date_commande, Adresse_livraison, Quantité)

Les Variétés sont des fleurs pour lesquelles nous précisons leur type, couleur et durée de vie. Les stocks précisent la date à laquelle une variété a été achetée et son nombre. Enfin, une personne peut acheter une certaine quantité de fleurs dans une commande à une date donnée.

Donner le schéma conceptuel entités/associations correspondant à ce schéma relationnel, en précisant si vous avez employé la notation UML ou MERISE (2 points).

Pour chacune des questions suivantes, donner l'expression algébrique et/ou la requête en SQL (en fonction de ce qui est demandé) permettant d'y répondre :

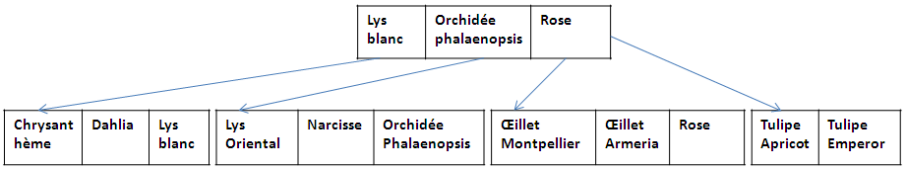
1. **(Algèbre + SQL)** Pour quelles variétés de couleur rouge au moins 31 fleurs sont disponibles en stock ? (1 point)
   Solution :
   $\pi_{Num\_Variete}(\sigma_{Couleur='rouge' \wedge Nombre\_fleurs \geq 31}(Variete \bowtie Stock)$
   SELECT Num_Variete
   FROM Variété v, Stock s
   WHERE Couleur='rouge' AND Nombre_fleurs ≥ 31 AND v.Num_variété = s.Num_Variété ;
2. **(Algèbre + SQL)** Pour quels stocks (Num_Variété, Date_Arrivée) la période de fraîcheur est dépassée ? (2 points)
   Solution :
   $\pi_{Num\_Variete, Date\_Arrivee}(Stock \bowtie_{NOW()-Date\_arrivee > Periode\_fraicheur \wedge Num\_variete = var} \rho_{Num\_variete \rightarrow var}(Variete))$
   SELECT Num_Variété, Date_Arrivée
   FROM Stock s, Variété v
   WHERE (NOW() - Date_arrivée) > Période_Fraicheur AND s.Num_variété = v.Num_variété ;
3. **(SQL)** Quelle variété a été la plus vendue ? Indiquer également le nombre de fleurs vendues pour cette variété. (2,5 points)
   Solution :
   SELECT Num_Variété, SUM(Quantité)
   FROM Commande
   GROUP BY Num_Variété
   HAVING SUM(Quantité) = (SELECT MAX(SUM(Quantité)) FROM Commande GROUP BY Num_variété) ;
4. **(Algèbre + SQL)** Quel client a commandé au moins une fois des fleurs de chaque variété ? (2,5 points)
   Solution :
   $(\pi_{Num\_Personne, Num\_Variete}(Commande) \div \pi_{Num\_Variete}(Variete)$
   SELECT Num_Personne
   FROM Commande
   WHERE Num_Personne NOT IN (

SELECT DISTINCT Num_Personne
FROM Commande
WHERE (Num_Personne, Num_Variété) NOT IN (
SELECT Num_Personne, Num_Variété FROM Personne, Variété)) ;

2 Organisation physique et optimisation (5 points)

Un index de type arbre B^+ est construit sur l'attribut Nom_Variété de la relation Variété. L'ordre d'insertion des enregistrements dans la table Variété correspond à l'ordre suivant des valeurs de l'attribut Nom_Variété : {Chrysanthème, Orchidée_phalaenopsis, Tulipe_Apricot, Dahlia, Rose, Iris_Sibérie, Lys_blanc, Oeillet_Montpellier, Lys_oriental, Tulipe_Emperor, Narcisse, Oeillet_armeria}.

1. Construire l'arbre B^+ **d'ordre 2** correspondant à cet ensemble de valeurs pour l'attribut Nom_Variété, en respectant l'ordre d'arrivée. Donner les étapes de construction intermédiaires importantes (celles qui produisent un changement de la structure de l'arbre). (1,5 points)
   Solution :



2. Donner un plan d'exécution sous forme d'arbre OU Explain d'Oracle pour la requête suivante, en considérant que le seul index disponible est sur l'attribut Nom_Variété de la relation Variété. Expliquer. (2 points)

   SELECT Nom, Prénom
   FROM Personne, Commande, Variété
   WHERE Personne.Num_Personne = Commande.Num_Personne
   AND Nom_Variété = 'Rose'
   AND Date_commande $\geq$ '2010-01-01' AND Date_commande $\leq$ '2010-01-09' ;
   Solution :

```
0  SELECT STATEMENT
1*   NESTED LOOPS
2*     NESTED LOOPS
3        TABLE ACCESS FULL    Personne
4*       TABLE ACCESS FULL    Commande
5      TABLE ACCESS BY ROWID  Variété
6*       INDEX RANGE SCAN     IDX_NOM_VARIETE

(1) Variété.Num_Variété = Commande.Num_Variété
(2) Commande.Num_Personne = Personne.Num_Personne
(4) Date_Commande between '2010-01-01' AND '2010-01-09'
(6) Nom_Variété = 'Rose'
```

3. Donner le plan d'exécution sous forme d'arbre OU Explain d'Oracle pour cette même requête, en considérant qu'il existe en plus un index sur l'attribut Date_commande de la relation Commande. Expliquez. (1,5 points)

Solution :

```
0  SELECT STATEMENT
1*   NESTED LOOPS
2*     NESTED LOOPS
3        TABLE ACCESS BY ROWID  Variété
4*         INDEX RANGE SCAN     IDX_NOM_VARIETE
5        TABLE ACCESS BY ROWID  Commande
6*         INDEX RANGE SCAN     IDX_NOM_VARIETE
7      TABLE ACCESS FULL        Personne

(1) Variété.Num_Variété = Commande.Num_Variété
(2) Commande.Num_Personne = Personne.Num_Personne
(4) Nom_Variété = 'Rose'
(5) Date_Commande between '2010-01-01' AND '2010-01-09'
```

3 Concurrence (5 points)

Soit l'exécution concurrente suivante :
H : $r_1[x]w_2[x]w_2[y]r_1[y]c_2w_1[x]w_1[y]c_1$

1. (1,5 points) Identifier les conflits et donner le graphe de sérialisation. Cette histoire est-elle sérialisable ?
   Solution :
   Conflits : sur x : $r_1[x] - w_2[x], w_2[x] - w_1[x]$, *sur y :* $w_2[y] - r_1[y], r_1[y] - w_1[y]$
   Le graphe de sérialisation contient un cycle T1 - T2, l'histoire (l'exécution) n'est donc pas sérialisable.
2. (1,5 points) Cette histoire évite-elle les annulations en cascade ? Afin de répondre à cette question, regarder quel est l'impact de la terminaison de T1 par Abort au lieu de Commit. Considérer ensuite le cas où T2 se termine par Abort au lieu de Commit.
   Solution :
   Elle n'évite pas les annulations en cascade. En effet, l'annulation de T2, provoquerait l'annulation de la lecture $r_1[y]$ avant le a_2.
3. (2 points) Trouver l'exécution produite par verrouillage à deux phases simple si les verrous d'une transaction sont toujours relâchés après sa validation ou son annulation. Les opérations bloquées en attente de verrou s'exécutent en priorité quand le verrou devient disponible, en respectant l'ordre de blocage. Dans le cas d'interblocage, les transactions bloquées seront annulées et non réinsérées.
   Solution :
   $r_1[x]$ *s'exécute en prenant le verrou de lecture sur x ($VL_1(x)$)*
   $w_2[x]$ *est bloqué par le verrou de lecture sur x*
   $w_2[y]$ *est bloqué car T2 est bloqué*
   $r_1[y]$ *s'exécute en prenant le verrou de lecture sur y ($VL_1(y)$)*
   c_2 *est bloqué car T2 est bloqué*
   $w_1[x]$ *s'exécute en prenant le verrou d'écriture sur x ($VE_1(x)$)*
   $w_1[y]$ *s'exécute en prenant le verrou d'écriture sur y ($VE_1(y)$)*
   c_1 *s'exécute et relâche les deux verrous en écriture*
   $w_2[x]$ *s'exécute en prenant le verrou d'écriture sur x ($VE_2(x)$)*
   $w_2[y]$ *s'exécute en prenant le verrou d'écriture sur y ($VE_2(y)$)*
   c_2 *s'exécute et relâche les deux verrous en écriture*
   Résultat : $r_1[x]r_1[y]w_1[x]w_1[y]c_1w_2[x]w_2[y]c_2$