



## Commission d'unité Hygiène et Sécurité " 07-2020 "

### EPE / SPA - DIVINDUS APMC - Unité Hennaya

#### Commission Jdida modifier

Libellé	Nombre
Nombre de réunions CHS	5
Nombre de réunions CHS extraordinaires	2
Nombre d'enquêtes menées par la CHS	1
Nombre de cas de recours à un expert	2

# Test 2 de TP

L'utilisation du SQL Developer est fortement recommandée

## Schéma

Salle (NumSalle, NomSalle)

Logiciel (NumLog, NomLog, DateAchat)



Poste (NumPoste, NomPoste, #NSalle)

Installer (#NumPoste, #NumLog)

## Questions

Donner les requêtes SQL qui permettent de :

- 1) Afficher le contenu de chaque table. **(2 pts)**
- 2) Donner le dernier logiciel acheté. **(3 pts)**
- 3) Trouver la salle qui ne contient aucun poste. **(4 pts)**
- 4) Montrer le poste dont tous les logiciels sont installés. **(4 pts)**
- 5) Trouver les noms des postes et les noms de salle où le logiciel « Open Office 4.1.3 » est installé. **(4 pts)**
- 6) Donner les numéros de poste où sont installés plus de 2 logiciels. **(3 pts)**

	<p style="text-align: center;">ANNÉE UNIVERSITAIRE 2009/2010 SESSION 1 DE PRINTEMPS</p> <p><b>PARCOURS : CSB4 &amp; CSB6</b>  <b>UE : INF 159, Bases de données</b>  <b>Épreuve : INF 159 EX</b>  <b>Date : Jeudi 6 mai 2010</b>  <b>Heure : 8 heures 30</b> <span style="float: right;"><b>Durée : 1 heure 30</b></span>  Documents : non autorisés  Épreuve de M. Alain GRIFFAULT</p>	
<p style="text-align: center;">DISVE Licence</p>		

## SUJET + CORRIGE

### Avertissement

- La plupart des questions sont indépendantes.
- Le barème total est de 23 points car le sujet est assez long.
- Le barème de chaque question est (approximativement) proportionnel à sa difficulté.
- L'espace pour répondre est suffisant (sauf si vous l'utilisez comme brouillon, ce qui est fortement déconseillé).

### Exercice 1 (SQL et normalisation (16 points))

*L'exercice porte sur une gestion simplifiée de groupes d'étudiants et d'enseignants. Chaque année universitaire, un étudiant est placé dans un groupe qui pour une matière donnée, a un professeur unique. Les professeurs n'interviennent que dans une seule matière tout au long de leur carrière.*

*Soit la relation Cours (Annee, Etudiant, Groupe, Professeur, Matière) et ses dépendances fonctionnelles :*

- $\{Annee, Etudiant\} \longrightarrow \{Groupe\}$  qui indique que chaque année, un étudiant appartient à un seul groupe.
- $\{Groupe, Matière\} \longrightarrow \{Professeur\}$  qui indique qu'un seul professeur est affecté à un groupe pour une matière donnée.
- $\{Professeur\} \longrightarrow \{Matière\}$  qui indique qu'un professeur n'enseigne qu'une seule matière.

**Question 1.1 (1 point)** *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les Professeur ayant travaillé en "2007-2008" pour le groupe "CSB6A12".*

**Réponse :**

$$R = \pi[Professeur](\sigma[Annee = 2007-2008 \wedge Groupe = CSB6A12](Cours))$$

-- les Professeur ayant travaillé en '2007-2008' pour le groupe 'CSB6A12'.

```
SELECT Professeur
FROM Cours
WHERE Annee = '2007-2008'
AND Groupe = 'CSB6A12';
```

**Question 1.2 (1 point)** *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les Professeur ayant enseigné dans au moins deux Groupe différents pour un même Etudiant.*

**Réponse :**

$$R = \pi[C1.Professeur](\sigma[ \begin{array}{l} C1.Professeur = C2.Professeur \\ \wedge C1.Etudiant = C2.Etudiant \\ \wedge C1.Groupe \neq C2.Groupe \end{array} ](\alpha[Cours : C1] \times \alpha[Cours : C2]))$$

-- Algèbre relationnelle

```
SELECT DISTINCT C1.Professeur
FROM Cours AS C1, Cours AS C2
WHERE C1.Professeur = C2.Professeur
AND C1.Etudiant = C2.Etudiant
AND C1.Groupe <> C2.Groupe;
-- Calcul relationnel
```

```

SELECT DISTINCT Professeur
FROM      Cours AS C1
WHERE EXISTS (SELECT *
              FROM    Cours AS C2
              WHERE    C1.Professeur = C2.Professeur
              AND      C1.Etudiant = C2.Etudiant
              AND      C1.Groupe <> C2.Groupe);

-- Calcul relationnel
SELECT DISTINCT Professeur
FROM      Cours AS C1
WHERE NOT (Groupe = ALL (SELECT Groupe
                        FROM    Cours AS C2
                        WHERE    C1.Professeur = C2.Professeur
                        AND      C1.Etudiant = C2.Etudiant));

-- Utilisation des agregas
SELECT DISTINCT Professeur, COUNT(Groupe)
FROM (SELECT DISTINCT Professeur, Etudiant, Groupe FROM Cours) AS R
GROUP BY      Professeur, Etudiant
HAVING        COUNT(Groupe) > 1;

```

**Question 1.3 (1 point)** Écrire une requête SQL qui caractérise les *Etudiant* ayant travaillé la même *Matiere* au moins deux *Annee* différentes.

**Réponse :**

```

-- Algebre relationnelle
SELECT DISTINCT C1.Etudiant
FROM      Cours AS C1, Cours AS C2
WHERE     C1.Etudiant = C2.Etudiant
        AND      C1.Matiere = C2.Matiere
        AND      C1.Annee <> C2.Annee;

-- Calcul relationnel
SELECT DISTINCT Etudiant
FROM      Cours AS C1
WHERE EXISTS (SELECT *
              FROM    Cours AS C2
              WHERE    C1.Etudiant = C2.Etudiant
              AND      C1.Matiere = C2.Matiere
              AND      C1.Annee <> C2.Annee);

-- Calcul relationnel
SELECT DISTINCT Etudiant
FROM      Cours AS C1
WHERE NOT (Annee = ALL (SELECT Annee
                        FROM    Cours AS C2
                        WHERE    C1.Etudiant = C2.Etudiant
                        AND      C1.Matiere = C2.Matiere));

-- Utilisation des agregas
SELECT DISTINCT Etudiant, COUNT(Annee)
FROM (SELECT DISTINCT Etudiant, Matiere, Annee FROM Cours) AS R
GROUP BY      Etudiant, Matiere
HAVING        COUNT(Annee) > 1;

```

**Question 1.4 (1 point)** Écrire une requête SQL qui caractérise les *Groupe* ayant eu moins de 5 *Professeur* différents.

**Réponse :**

```

-- Utilisation des agregas
SELECT DISTINCT Groupe, COUNT(Professeur)
FROM (SELECT DISTINCT Groupe, Professeur FROM Cours) AS R
GROUP BY      Groupe
HAVING        COUNT(Professeur) < 5;

```

**Question 1.5 (2 points)** Écrire une requête SQL qui caractérise les *Etudiant* ayant appartenu à un *Groupe* qui a eu moins de 2 *Professeur* différents, et n'ayant pas appartenu à un *Groupe* qui a eu plus de 9 *Professeur* différents.

**Réponse :**

```
-- Utilisation des agregas
SELECT DISTINCT Etudiant
FROM      Cours AS C,
          (SELECT DISTINCT Groupe, COUNT(Professeur)
           FROM (SELECT DISTINCT Groupe, Professeur FROM Cours) AS R
           GROUP BY      Groupe
           HAVING        COUNT(Professeur) < 2) AS Rabachage;
WHERE     C.Groupe = Rabachage.Groupe
EXCEPT
SELECT DISTINCT Etudiant
FROM      Cours AS C,
          (SELECT DISTINCT Groupe, COUNT(Professeur)
           FROM (SELECT DISTINCT Groupe, Professeur FROM Cours) AS R
           GROUP BY      Groupe
           HAVING        COUNT(Professeur) > 9) AS Decouverte;
WHERE     C.Groupe = Decouverte.Groupe
```

**Question 1.6 (2 points)** Traduisez l'expression algébrique suivante :

$$R = \pi_{[Annee, Groupe, Matiere]}(Cours) - \pi_{[C1.Annee, C1.Groupe, C1.Matiere]}(\sigma_{\begin{array}{l} C1.Annee \neq C2.Annee \\ \wedge C1.Groupe = C2.Groupe \\ \wedge C1.Matiere = C2.Matiere \end{array}}](\alpha[Cours : C1] \times \alpha[Cours : C2]))$$

en une requête SQL, puis expliquez ce qu'elle calcule.

**Réponse :**

```
-- Algebre relationnelle
SELECT DISTINCT Annee, Groupe, Matiere
FROM      Cours
EXCEPT
SELECT C1.Annee, C1.Groupe, C1.Matiere
FROM      Cours AS C1, Cours AS C2
WHERE     C1.Annee <> C2.Annee
AND       C1.Groupe = C2.Groupe
AND       C1.Matiere = C2.Matiere;
-- Calcul relationnel
SELECT DISTINCT Annee, Groupe, Matiere
FROM      Cours AS C1
WHERE     NOT EXISTS
          (SELECT *
           FROM      Cours AS C2
           WHERE     C1.Annee <> C2.Annee
           AND       C1.Groupe = C2.Groupe
           AND       C1.Matiere = C2.Matiere);
```

La requête SQL caractérise les couples (*Groupe*, *Matiere*) spécifiques à une *Annee*.

**Question 1.7 (2 points)** Écrire une requête SQL qui caractérise les *Etudiant* ayant eu tous les *Professeur*.

**Réponse :**

```
-- Algebre relationnelle
SELECT DISTINCT Etudiant
FROM      Cours
EXCEPT
SELECT Etudiant
```

```

FROM (
  SELECT *
  FROM (SELECT DISTINCT Etudiant FROM Cours) AS P1R1,
       (SELECT DISTINCT Professeur FROM Cours) AS R2
  EXCEPT
  SELECT Etudiant, Professeur
  FROM Cours
) AS NonEntierR1;
-- Calcul relationnel
SELECT DISTINCT Etudiant
FROM Cours AS P1
WHERE NOT EXISTS
  (SELECT DISTINCT Professeur
   FROM Cours AS P2
   WHERE NOT EXISTS
     (SELECT DISTINCT Professeur
      FROM Cours AS P11
      WHERE P11.Etudiant = P1.Etudiant
      AND P11.Professeur = P2.Professeur));

```

Les questions suivantes portent sur la normalisation de la relation *Cours*.

**Question 1.8 (1 point)** Donnez toutes les clefs candidates de la relation *Cours*.

**Réponse :**

– Les dépendances fonctionnelles donnent :  $C_1 = \{\text{Etudiant}, \text{Annee}, \text{Professeur}\}$  et  $C_2 = \{\text{Etudiant}, \text{Annee}, \text{Matiere}\}$

**Question 1.9 (1 point)** Même si l'on suppose qu'il n'y a aucun doublon dans *Cours*, justifiez pourquoi la relation *Cours* n'est pas en troisième forme normale.

**Réponse :** Une seule des explications suivantes est suffisante (liste non exhaustive).

**Non 2NF :** La clef  $\{\text{Etudiant}, \text{Annee}, \text{Professeur}\}$  contient  $\{\text{Professeur}\}$  qui détermine  $\{\text{Matiere}\}$ .

**Non 2NF :** La clef  $\{\text{Etudiant}, \text{Annee}, \text{Professeur}\}$  contient  $\{\text{Etudiant}, \text{Annee}\}$  qui détermine  $\{\text{Groupe}\}$ .

**Question 1.10 (2 points)** Appliquez un algorithme (ou une technique) de normalisation pour obtenir une décomposition, sans perte d'information et sans perte de dépendance fonctionnelle, de la relation *Cours* en un ensemble de relations en troisième forme normale. Vous n'écrirez sur la copie que les nouvelles relations et les dépendances fonctionnelles qui sont à la base des projections effectuées.

**Réponse :** Décomposition en 3NF :

1.  $\{\text{Etudiant}, \text{Annee}\} \longrightarrow \{\text{Groupe}\}$  donne *Inscrits* (*Etudiant*, *Annee*, *Groupe*) 3NF et BCNF.
2.  $\{\text{Groupe}, \text{Matiere}\} \longrightarrow \{\text{Professeur}\}$  donne *Repartition* (*Groupe*, *Matiere*, *Professeur*) 3NF et non BCNF.

**Question 1.11 (2 points)** Après avoir précisé si votre décomposition est en BCNF ou bien seulement en 3NF, répondez à la question qui vous concerne.

**Votre décomposition est en BCNF :**

– Indiquez la dépendance fonctionnelle que vous avez perdue.

**Votre décomposition est seulement en 3NF :**

– Indiquez le problème de redondance qui subsiste.

**Réponse :**

**Décomposition en BCNF :**

1.  $\{\text{Groupe}, \text{Matiere}\} \longrightarrow \{\text{Professeur}\}$  qui indique qu'un seul professeur est affecté à un groupe pour une matière donnée.

**Votre décomposition est seulement en 3NF :**

1. L'information (*Professeur*, *Matiere*) est dupliquée.

**Exercice 2 (Évitement de l'interblocage (4 points))**

La sérialisation des transactions est souvent obtenue à l'aide de verrous. Un verrou est un triplet (état du verrou ( $L, S$  ou  $X$ ), liste des détenteurs du verrou, liste des demandes). Un exemple classique d'interblocage lors d'un verrouillage strict avec deux types de verrous est :

Transaction A	temps	Transaction B	Verrou(tuple)
	$t0$		$(L, \emptyset, \emptyset)$
$dem(select(tuple))$	$t1.1$		$(L, \emptyset, \{lecture(A)\})$
$select(tuple)$	$t1.2$		$(S, \{A\}, \emptyset)$
	$t2.1$	$dem(select(tuple))$	$(S, \{A\}, \{lecture(B)\})$
	$t2.2$	$select(tuple)$	$(S, \{A, B\}, \emptyset)$
$dem(update(tuple))$	$t3.1$		$(S, \{A, B\}, \{écriture(A)\})$
	$t4.1$	$dem(update(tuple))$	$(S, \{A, B\}, \{écriture(A), écriture(B)\})$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

L'évitement consiste à adapter le protocole à deux phases en mémorisant pour chaque transaction une estampille qui est sa date de création. Cette estampille sert pour soit tuer une transaction, soit s'auto-détruire. Deux versions lorsque  $(T_i, e_i)$  demande un verrou sur  $tuple_j$  détenu par  $(T_k, e_k)$ .

**Wait-Die :** si  $e_i < e_k$ ,  $T_i$  attend, sinon  $T_i$  meurt.

**Wound-Wait :** si  $e_i < e_k$ ,  $T_i$  blesse  $T_k$ , sinon  $T_i$  attend.

Dans les deux cas, la transaction tuée redémarre plus tard en gardant son estampille d'origine.

**Question 2.1 (4 points)** Compléter le tableau suivant en utilisant la version **Wound-Wait** de l'évitement. Les transactions doivent se terminer par un **COMMIT** après leur update réussi.

**Réponse :**

<i>Transaction A</i>	<i>temps</i>	<i>Transaction B</i>	<i>Verrou(tuple)</i>
	<i>t0</i>		$(L, \emptyset, \emptyset)$
<i>dem(select(tuple))</i>	<i>t1.1</i>		$(L, \emptyset, \{lecture(A, t1)\})$
<i>select(tuple)</i>	<i>t1.2</i>		$(S, \{(A, t1)\}, \emptyset)$
	<i>t2.1</i>	<i>dem(select(tuple))</i>	$(S, \{(A, t1)\}, \{lecture(B, t2)\})$
	<i>t2.2</i>	<i>select(tuple)</i>	$(S, \{(A, t1), (B, t2)\}, \emptyset)$
<i>dem(update(tuple))</i>	<i>t3.1</i>		$(S, \{(A, t1), (B, t2)\}, \{écriture(A, t1)\})$
	<i>t4.1</i>	<i>dem(rollback)</i>	$(S, \{(A, t1), (B, t2)\}, \{écriture(A, t1), rollback(B, t2)\})$
	<i>t4.2</i>	<i>rollback</i>	$(S, \{(A, t1)\}, \{écriture(A, t1)\})$
<i>update(tuple)</i>	<i>t3.2</i>		$(X, \{(A, t1)\}, \emptyset)$
<i>dem(commit)</i>	<i>t5.1</i>		$(X, \{(A, t1)\}, \{commit(A, t1)\})$
<i>commit</i>	<i>t5.2</i>		$(L, \emptyset, \emptyset)$
	<i>t6.1</i>	<i>dem(select(tuple))</i>	$(L, \emptyset, \{lecture(B, t2)\})$
	<i>t6.2</i>	<i>select(tuple)</i>	$(S, \{(B, t2)\}, \emptyset)$
	<i>t7.1</i>	<i>dem(update(tuple))</i>	$(S, \{(B, t2)\}, \{écriture(B, t2)\})$
	<i>t7.2</i>	<i>(update(tuple))</i>	$(X, \{(B, t2)\}, \emptyset)$
	<i>t8.1</i>	<i>dem(commit)</i>	$(X, \{(B, t2)\}, \{commit(B, t2)\})$
	<i>t8.2</i>	<i>(commit)</i>	$(L, \emptyset, \emptyset)$
	<i>t9</i>		$(L, \emptyset, \emptyset)$

**Exercice 3 (Optimisation des requêtes (3 points))**

**Question 3.1 (1 point)** Donner les objectifs principaux des modules d'optimisation des requêtes présents dans les SGBD.

**Réponse :**

1. Diminuer le temps de calcul des requêtes.
2. Laisser le choix aux développeurs d'applications pour l'écriture des requêtes.
3. Ne pas avoir à informer les développeurs d'applications des choix de structures de données retenus.

**Question 3.2 (2 points)** Donner les techniques de bases de l'optimisation des requêtes et en quelques mots leurs avantages et inconvénients.

**Réponse :**

1. Réécriture de requêtes.
2. Compression des données.
3. Indexation, hachage.
4. Statistiques sur les tailles des relations.
5. Utilisation des dépendances fonctionnelles.