| | |
|---|---|
| Département Licence | ANNÉE : 2006/2007 — 2$^{ème}$ SESSION SEMESTRE D'AUTOMNE<br>**PARCOURS :** INF 3<br>**UE :** INF 109, Bases de données<br>**Épreuve :** INF 109 EX<br>**Date :**<br>**Heure :**<br>**Durée :** 1 heure 30<br>Documents : non autorisés<br>Épreuve de M. Alain GRIFFAULT |

# SUJET + CORRIGE

## Avertissement

- La plus part des questions sont indépendantes.
- Le barème total est de 22 points car le sujet est assez long.
- Le barème est (approximativement) proportionnel à la difficulté de la question.
- L'espace laissé pour les réponses est suffisant (sauf si vous utilisez ces feuilles comme brouillon, ce qui est fortement déconseillé).

## Le sujet

Le sujet porte sur une gestion de la construction d'un lotissement. Un lotissement est constitué de plusieurs maisons. La construction d'une maison est découpée en tâches (fondations, toiture, finitions, ...), plusieurs métiers étant utiles à la réalisation d'une même tâche.
Soit `Planning (Maison, PlanType, NbJours, Jour, Tache, Metier, Artisan)` une relation et ses dépendances fonctionnelles :

- {`Maison`} ⟶ {`PlanType, NbJours`} qui indique qu'une maison détermine son plan type et le nombre de jours nécessaire à sa construction.
- {`PlanType`} ⟶ {`NbJours`} qui indique que le nombre de jours nécessaire à la construction, est fonction du plan type de la maison.
- {`Artisan`} ⟶ {`Metier`} qui indique qu'un artisan n'a qu'un seul métier.
- {`Maison, Jour`} ⟶ {`Tache, Metier`} qui indique que le découpage par métier d'une tâche est journalier pour une maison.
- {`Tache, Metier`} ⟶ {`Artisan`} qui indique que pour toutes les maisons, tous les travaux liés à un métier pour une tâche donnée sont réalisés par le même artisan.

La principale simplification concerne le fait qu'il n'y a jamais simultanément plusieurs artisans dans une maison.
Par exemple :

| Maison | PlanType | NbJours | Jour | Tache | Metier | Artisan |
|---|---|---|---|---|---|---|
| Num 5 | T5bis | 120 | 28 fevrier 2007 | Fondations | Maçon | Bouygues |

spécifie que le 28 février 2007, l'artisan Bouygues travaillera aux fondations. du T5bis "Num 5", maison pour laquelle il est prévu 120 jours de travaux.

**Exercice 1 (Requêtes SQL sur *Planning* (11 points))**

***Question 1.1 (1 point)*** *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les* ***`Artisan`*** *ayant travaillé aux* ***`Fondations`*** *d'une maison de type* ***`T5bis`****.*

**Réponse :**

$$R = \pi[\mathtt{Artisan}](\sigma[\mathtt{Tache} = \mathtt{Fondations} \wedge \mathtt{PlanType} = \mathtt{T5bis}](\mathtt{Planning}))$$

```
SELECT DISTINCT Artisan
FROM            Planning
WHERE           Tache='Fondations'
  AND           PlanType='T5bis';
```

***Question 1.2 (1 point)*** *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les* **Artisan** *ayant travaillé à au moins deux* **Tache** *différentes pour une même* **Maison**.

**Réponse :**

$$R = \pi[\mathit{P1.Artisan}](\sigma[\ \mathit{P1.Artisan} = \mathit{P2.Artisan} \wedge \mathit{P1.Maison} = \mathit{P2.Maison} \wedge \mathit{P1.Tache} \neq \mathit{P2.Tache}\ ](\alpha[P : \mathit{P1}] \times \alpha[P : \mathit{P2}]))$$

```
-- Algebre relationnelle
SELECT DISTINCT P1.Artisan
FROM            Planning AS P1, Planning AS P2
WHERE           P1.Artisan = P2.Artisan
  AND           P1.Maison = P2.Maison
  AND           P1.Tache <> P2.Tache;
-- Calcul relationnel
SELECT DISTINCT Artisan
FROM            Planning AS P1
WHERE EXISTS    (SELECT *
                 FROM   Planning AS P2
                 WHERE  P1.Artisan = P2.Artisan
                 AND    P1.Maison = P2.Maison
                 AND    P1.Tache <> P2.Tache);
-- Calcul relationnel
SELECT DISTINCT Artisan
FROM            Planning AS P1
WHERE NOT (Tache = ALL (SELECT Tache
                        FROM   Planning AS P2
                        WHERE  P1.Artisan = P2.Artisan
                          AND  P1.Maison = P2.Maison));
-- Utilisation des agregas
SELECT DISTINCT Artisan, COUNT(Tache)
FROM (SELECT DISTINCT Artisan, Maison, Tache FROM Planning) AS R
GROUP BY        Artisan, Maison
HAVING          COUNT(Tache) > 1;
```

***Question 1.3 (1 point)*** *Écrire une requête SQL qui caractérise les* **Artisan** *ayant travaillé le même* **Jour** *sur au moins deux* **Maison** *différentes.*

**Réponse :**

```
-- Algebre relationnelle
SELECT DISTINCT P1.Artisan
FROM            Planning AS P1, Planning AS P2
WHERE           P1.Artisan = P2.Artisan
  AND           P1.Jour = P2.Jour
  AND           P1.Maison <> P2.Maison;

-- Calcul relationnel
SELECT DISTINCT Artisan
FROM            Planning AS P1
WHERE EXISTS    (SELECT *
                 FROM   Planning AS P2
                 WHERE  P1.Artisan = P2.Artisan
                 AND    P1.Jour = P2.Jour
                 AND    P1.Maison <> P2.Maison);
-- Calcul relationnel
SELECT DISTINCT Artisan
FROM            Planning AS P1
WHERE NOT (Maison = ALL (SELECT Maison
                         FROM   Planning AS P2
                         WHERE  P1.Artisan = P2.Artisan
                         AND    P1.Jour = P2.Jour));
```

```
-- Utilisation des agregas
SELECT DISTINCT Artisan, COUNT(Maison)
FROM (SELECT DISTINCT Artisan, Jour, Maison FROM Planning) AS R
GROUP BY        Artisan, Jour
HAVING          COUNT(Maison) > 1;
```

***Question 1.4 (1 point)*** *Donnez une contrainte SQL entre les attributs afin que la requête précédente retourne toujours un ensemble vide.*

**Réponse :**

```
UNIQUE(Artisan,Jour),
```

***Question 1.5 (1 point)*** *Écrire une requête SQL qui caractérise les* **`Maison`** *dont la durée de construction a dépassé d'au moins 10 jours le* **`NbJours`** *prévu.*

**Réponse :**

```
-- Utilisation des agregas
SELECT DISTINCT Maison, NbJours, COUNT(Jour)
FROM (SELECT DISTINCT Maison, NbJours, Jour FROM Planning) AS R
GROUP BY        Maison, NbJours
HAVING          COUNT(Jour) > NbJours + 9;
```

***Question 1.6 (2 points)*** *Écrire une requête SQL qui caractérise les* **`Artisan`** *ayant travaillé à une* **`Maison`** *dont la construction a été trop longue d'au moins 10 jours, et n'ayant pas travaillé à une* **`Maison`** *dont la durée de construction a été d'au moins 10 jours plus courte que celle prévue.*

**Réponse :**

```
-- Utilisation des agregas
SELECT DISTINCT Artisan
FROM            Planning AS P,
                (SELECT DISTINCT Maison
                 FROM (SELECT DISTINCT Maison, NbJours, Jour FROM Planning)AS R
                 GROUP BY        Maison, NbJours
                 HAVING          COUNT(Jour) > NbJours + 9) AS Retard
WHERE           P.Maison = Retard.Maison
  EXCEPT
SELECT DISTINCT Artisan
FROM            Planning AS P,
                (SELECT DISTINCT Maison
                 FROM (SELECT DISTINCT Maison, NbJours, Jour FROM Planning)AS R
                 GROUP BY        Maison, NbJours
                 HAVING          COUNT(Jour) < NbJours - 9) AS Avance
WHERE           P.Maison = Avance.Maison;
```

***Question 1.7 (2 points)*** *Traduisez l'expression algébrique suivante dans laquelle* **`P`** *est une abréviation de* **`Planning`**.

$$\begin{aligned} R \;=\; & \pi[\mathtt{PlanType,Tache,Metier}](P) - \\ & \pi[\mathtt{P1.PlanType,P1.Tache,P1.Metier}](\sigma[\; \begin{array}{l} \mathtt{P1.PlanType} \neq \mathtt{P2.PlanType} \\ \wedge\ \mathtt{P1.Tache} = \mathtt{P2.Tache} \\ \wedge\ \mathtt{P1.Metier} = \mathtt{P2.Metier} \end{array} \;](\alpha[\mathtt{P\ :P1}] \times \alpha[\mathtt{P\ :P2}])) \end{aligned}$$

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

**Réponse :**

```
-- Algebre relationnelle
SELECT DISTINCT PlanType, Tache, Metier
FROM   Planning
  EXCEPT
SELECT P1.PlanType, P1.Tache, P1.Metier
FROM   Planning AS P1, Planning AS P2
```

```
WHERE  P1.PlanType <> P2.PlanType
AND    P1.Tache = P2.Tache
AND    P1.Metier = P2.Metier;
-- Calcul relationnel
SELECT DISTINCT PlanType, Tache, Metier
FROM   Planning AS P1
WHERE  NOT EXISTS
       (SELECT *
        FROM   Planning AS P2
        WHERE  P1.PlanType <> P2.PlanType
        AND    P1.Tache = P2.Tache
        AND    P1.Metier = P2.Metier);
```

*La requête SQL caractérise les couples* **`(Tache,Metier)`** *spécifiques à un* **`PlanType`**.

***Question 1.8 (2 points)*** *Écrire une requête SQL qui caractérise les* **`Artisan`** *ayant travaillé dans toutes les* **`Maison`**.

**Réponse :**

```
-- Algebre relationnelle
SELECT DISTINCT Artisan
FROM   Planning
  EXCEPT
SELECT  Artisan
FROM (
  SELECT *
  FROM   (SELECT DISTINCT Artisan FROM Planning) AS PiR1,
         (SELECT DISTINCT Maison FROM Planning) AS R2
    EXCEPT
  SELECT Artisan, Maison
  FROM   Planning
) AS NonEntierR1;
-- Calcul relationnel
SELECT DISTINCT Artisan
FROM   Planning AS P1
WHERE  NOT EXISTS
       (SELECT DISTINCT Maison
         FROM  Planning AS P2
         WHERE  NOT EXISTS
                (SELECT DISTINCT Maison
                 FROM   Planning AS P11
                 WHERE  P11.Artisan  = P1.Artisan
                 AND    P11.Maison  = P2.Maison));
```

**Exercice 2 (Normalisation (7 points))** *Le schéma conceptuel de la relation* **`Planning`** *comprend :*
– *les dépendances fonctionnelles définies dans le sujet.*
– *la contrainte définie à la question 1.4.*

***Question 2.1 (1 point)*** *Donnez toutes les clefs candidates de la relation* **`Planning`**.

**Réponse :**
– *Les dépendances fonctionnelles donnent :* $C_1 = \{$**`Maison, Jour`**$\}$
– *La contrainte donne :* $C_2 = \{$**`Artisan, Jour`**$\}$.

***Question 2.2 (1 point)*** *Même si l'on suppose qu'il n'y a aucun doublon dans* **`Planning`**, *justifiez pourquoi la relation* **`Planning`** *n'est pas en troisième forme normale.*

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

**Non 2NF :** *La clef* $\{$**`Maison, Jour`**$\}$ *contient* $\{$**`Maison`**$\}$ *qui détermine* $\{$**`PlanType`**$\}$.

**Non 3NF :** *La clef* $\{$**`Maison, Jour`**$\}$ *et* $\{$**`PlanType`**$\} \longrightarrow \{$**`NbJours`**$\}$.

**Non 3NF :** *La clef* $\{$**`Maison, Jour`**$\}$ *et* $\{$**`Artisan`**$\} \longrightarrow \{$**`Metier`**$\}$.

***Question 2.3 (2 points)*** *Appliquez un algorithme (ou une technique) de normalisation pour obtenir une décomposition, sans perte d'information, de la relation* ***Planning*** *en un ensemble de relations au moins 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 :*
- ***ListeMaisons (Maison, PlanType)*** *qui provient de* {***Maison***} ⟶ {***PlanType***}
- ***ListeTypesMaison (PlanType, NbJours)*** *qui provient de* {***PlanType***} ⟶ {***NbJours***}
- ***ListeTravaux3NF (Maison, Jour, Tache, Metier)*** *qui provient de* {***Maison, Jour***} ⟶ {***Tache, Metier***}
- ***AttributionMarche (Tache, Metier, Artisan)*** *qui provient de* {***Tache, Metier***} ⟶ {***Artisan***}

*Décomposition en BCNF :*
- ***ListeMaisons (Maison, PlanType)*** *qui provient de* {***Maison***} ⟶ {***PlanType***}
- ***ListeTypesMaison (PlanType, NbJours)*** *qui provient de* {***PlanType***} ⟶ {***NbJours***}
- ***ListeTravauxBCNF (Maison, Jour, Tache, Artisan)*** *qui provient de* {***Maison, Jour***} ⟶ {***Tache, Artisan***}
- ***ListeArtisans (Artisan, Metier)*** *qui provient de* {***Artisan***} ⟶ {***Metier***}

***Question 2.4 (3 points)*** *Après avoir précisé si votre décomposition est en BCNF ou bien seulement en 3NF, répondez aux questions qui vous concernent.*

**Votre décomposition est en BCNF** *:*
1. *Indiquez la dépendance fonctionnelle que vous avez perdue.*
2. *Indiquez comment s'exprime la contrainte de la question 1.4.*
3. *Donnez une requête d'insertion possible pour une de vos relations qui viole une dépendance fonctionnelle de la relation* ***Planning****.*

**Votre décomposition est seulement en 3NF** *:*
1. *Indiquez le problème de redondance qui subsiste.*
2. *Indiquez comment s'expriment la contrainte de la question 1.4.*
3. *Donnez une requête d'insertion qui nécessite que le SGBD évalue une contrainte inter-relations pour déterminer si elle est possible ou non.*

**Réponse :**

**Décomposition en BCNF** *:*
1. {***Tache, Metier***} ⟶ {***Artisan***} *qui indique que pour les travaux d'une tâche, le métier détermine l'artisan est perdue.*
2. *La contrainte* `UNIQUE(Artisan,Jour)` *est devenue une contrainte de la relation* ***ListeTravaux****.*
3. *Les insertions suivantes sont possibles :*

```
INSERT INTO ListeArtisans VALUES ('Bianchi', 'Bases de donnees');
INSERT INTO ListeArtisans VALUES ('Griffault', 'Bases de donnees');
INSERT INTO ListeTravauxBCNF VALUES ('113', '2007-1-20', 'GR1', 'Bianchi');
-- l'insertion suivante viole la df (Tache,Metier) -> Artisan
INSERT INTO ListeTravauxBCNF VALUES ('310', '2007-1-21', 'GR1', 'Griffault');
```

**Votre décomposition est seulement en 3NF** *:*
1. *L'information* `(Artisan, Metier)` *est dupliquée.*
2. *La contrainte* `UNIQUE(Artisan,Jour)` *est devenue une contrainte entre les relations* ***ListeTravaux*** *et* ***AttributionMarche****.*
3. *L'insertion suivante nécessite d'évaluer une contrainte inter-relations.*

```
INSERT INTO AttributionMarche VALUES ('GR1', 'Bases de donnees', 'Bianchi');
INSERT INTO AttributionMarche VALUES ('GR2', 'Bases de donnees', 'Bianchi');
INSERT INTO ListeTravaux3NF VALUES ('113', '2007-1-20', 'GR1', 'Bases de donnees');
-- l'insertion suivante viole la contrainte UNIQUE(Artisan, Jour)
INSERT INTO ListeTravaux3NF VALUES ('310', '2007-1-20', 'GR2', 'Bases de donnees');
```

## Exercice 3 (Évitement de l'interblocage (4 points))

***Définition 1 (interblocage)*** *Un ensemble de transactions est inter-bloqué lorsqu'il existe un cycle dans le graphe d'attente défini par :*
- *les arcs* $(T_i, tuple_j)$ *qui indique que* $T_i$ *attend un verrou sur* $tuple_j$*,*
- *les arcs* $(tuple_i, T_j)$ *qui indique que* $tuple_i$ *est détenu par* $T_j$*,*

*Un exemple classique est :*

| *Transaction A* | *temps* | *Transaction B* | *Verrou*($tuple1$) | *Verrou*($tuple2$) |
|---|---|---|---|---|
| $dem(update(tuple1))$ | *t1.1* | | $(L, \emptyset, \{ecriture(A)\})$ | $(L, \emptyset, \emptyset)$ |
| $update(tuple1)$ | *t1.2* | | $(X, \{A\}, \emptyset)$ | $(L, \emptyset, \emptyset)$ |
| | *t2.1* | $dem(update(tuple2))$ | $(X, \{A\}, \emptyset)$ | $(L, \emptyset, \{ecriture(B)\})$ |
| | *t2.2* | $update(tuple2)$ | $(X, \{A\}, \emptyset)$ | $(X, \{B\}, \emptyset)$ |
| | *t3.1* | $dem(update(tuple1))$ | $(X, \{A\}, \{ecriture(B)\})$ | $(X, \{B\}, \emptyset)$ |
| $dem(update(tuple2))$ | *t4.1* | | $(X, \{A\}, \{ecriture(B)\})$ | $(X, \{B\}, \{ecriture(A)\})$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |

*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 3.1 (4 points)*** *Compléter le tableau suivant en utilisant la version* **`Wound-Wait`** *de l'évitement. Les transactions doivent se terminer par un* **`COMMIT`**.

**Réponse :**

| *Transaction A* | *temps* | *Transaction B* | *Verrou*($tuple1$) | *Verrou*($tuple2$) |
|---|---|---|---|---|
| $dem(update(tuple1))$ | *t1.1* | | $(L, \emptyset, \{ecriture(A, t1)\})$ | $(L, \emptyset, \emptyset)$ |
| $update(tuple1)$ | *t1.2* | | $(X, \{(A, t1)\}, \emptyset)$ | $(L, \emptyset, \emptyset)$ |
| | *t2.1* | $dem(update(tuple2))$ | $(X, \{(A, t1)\}, \emptyset)$ | $(L, \emptyset, \{ecriture(B, t2)\})$ |
| | *t2.2* | $update(tuple2)$ | $(X, \{(A, t1)\}, \emptyset)$ | $(X, \{(B, t2)\}, \emptyset)$ |
| | *t3.1* | $dem(update(tuple1))$ | $(X, \{(A, t1)\}, \{ecriture(B, t2)\})$ | $(X, \{(B, t2)\}, \emptyset)$ |
| $dem(update(tuple2))$ | *t4.1* | | $(X, \{(A, t1)\}, \{ecriture(B, t2)\})$ | $(X, \{(B, t2)\}, \{ecriture(A, t1)\})$ |
| | *t5* | *ROLLBACK* | $(X, \{(A, t1)\}, \emptyset)$ | $(L, \emptyset, \{ecriture(A, t1)\})$ |
| $update(tuple2)$ | *t4.2* | | $(X, \{(A, t1)\}, \emptyset)$ | $(X, \{(A, t1)\}, \emptyset)$ |
| *COMMIT* | *t6* | | $(L, \emptyset, \emptyset)$ | $(L, \emptyset, \emptyset)$ |
| | *t7.1* | $dem(update(tuple2))$ | $(L, \emptyset, \emptyset)$ | $(L, \emptyset, \{ecriture(B, t2)\})$ |
| | *t7.2* | $update(tuple2)$ | $(L, \emptyset, \emptyset)$ | $(X, \{(B, t2)\}, \emptyset)$ |
| | *t8.1* | $dem(update(tuple1))$ | $(L, \emptyset, \{ecriture(B, t2)\})$ | $(X, \{(B, t2)\}, \emptyset)$ |
| | *t8.2* | $update(tuple1)$ | $(X, \{(B, t2)\}, \emptyset)$ | $(X, \{(B, t2)\}, \emptyset)$ |
| | *t9* | *COMMIT* | $(L, \emptyset, \emptyset)$ | $(L, \emptyset, \emptyset)$ |