Département Licence

Année : 2006/2007 — $2^{ème}$ session Semestre d'automne

PARCOURS : INF 3
UE : INF 109, Bases de données
Épreuve : INF 109 EX
Date :
Heure :
Durée : 1 heure 30
Documents : non autorisés
Épreuve de M. Alain Griffault

Code d'anonymat :

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`} $\longrightarrow$ {`PlanType, NbJours`} qui indique qu'une maison détermine son plan type et le nombre de jours nécessaire à sa construction.
- {`PlanType`} $\longrightarrow$ {`NbJours`} qui indique que le nombre de jours nécessaire à la construction, est fonction du plan type de la maison.
- {`Artisan`} $\longrightarrow$ {`Metier`} qui indique qu'un artisan n'a qu'un seul métier.
- {`Maison, Jour`} $\longrightarrow$ {`Tache, Metier`} qui indique que le découpage par métier d'une tâche est journalier pour une maison.
- {`Tache, Metier`} $\longrightarrow$ {`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`****.*

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`**.

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

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

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.*

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.*

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

$$
\begin{array}{rcl}
R & = & \pi[\texttt{PlanType},\texttt{Tache},\texttt{Metier}](P) - \\
 & & \pi[\texttt{P1.PlanType},\texttt{P1.Tache},\texttt{P1.Metier}](\sigma[\ \texttt{P1.PlanType} \neq \texttt{P2.PlanType} \\
 & & \qquad\qquad \wedge\ \texttt{P1.Tache} = \texttt{P2.Tache} \\
 & & \qquad\qquad \wedge\ \texttt{P1.Metier} = \texttt{P2.Metier} \quad\ ](\alpha[\texttt{P : P1}] \times \alpha[\texttt{P : P2}]))
\end{array}
$$

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

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

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.*

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.*

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.*

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`*.*

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	$dem(update(tuple1))$		