DISVE Licence	Année Universitaire 2009/2010 Session 2 de Printemps **PARCOURS :** CSB4 & CSB6 **UE :** INF 159, Bases de données **Épreuve :** INF 159 EX **Date :** Mardi 22 juin 2010 **Heure :** 8 heures 30 **Durée :** 1 heure 30 Documents : non autorisés Épreuve de M. Alain Griffault	

Code d'anonymat :

Avertissement
- 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))

Un informaticien souhaite utiliser un SGBD relationnel pour la gestion de ses fichiers source. Chaque fichier peut avoir plusieurs versions, chacune de ces versions étant utilisée pour une bibliothèque ou bien pour une application (que l'on regroupe sous le terme logiciel). Il compile ses applications avec les bibliothèques nécessaires sur plusieurs plate-formes afin d'obtenir des binaires exécutables sur différents processeurs.

L'informaticien s'impose des règles pour simplifier sa gestion. Les informations stockées sont celles de la relation `Codes (Fichier, Version, Logiciel, Processeur, Binaire)` *et respectent les dépendances fonctionnelles :*
- {`Version, Fichier`} ⟶ {`Logiciel`} *: chaque version d'un fichier n'est utilisée que dans un seul logiciel.*
- {`Logiciel, Processeur`} ⟶ {`Binaire`} *qui indique qu'un logiciel (bibliothèque ou application) sur un processeur donné n'est utilisé que dans un seul binaire.*
- {`Binaire`} ⟶ {`Processeur`} *qui indique qu'un binaire ne peut s'exécuter que sur un seul type de processeur.*

Voici un exemple de tuples possibles dans `Codes`*.*

Fichier	*Version*	*Logiciel*	*Processeur*	*Binaire*
fic1.c	*V1*	*A1*	*i386*	*B1-i386*
fic2.c	*V3*	*A1*	*SPARC*	*B1-SPARC*
fic3.c	*V2*	*L1*	*i386*	*B1-i386*
fic3.c	*V2*	*L1*	*SPARC*	*B1-SPARC*

Question 1.1 (1 point) *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les* `Binaire` *utilisant une version "V43" pour le logiciel "L16".*

Question 1.2 (1 point) *Après en avoir donné une écriture algébrique, écrire une requête SQL qui caractérise les* `Binaire` *utilisant au moins deux* `Logiciel` *différents, et tels que deux fichiers aient les mêmes* `Version`*.*

Question 1.3 (1 point) *Écrire une requête SQL qui caractérise les* `Fichier` *compilés sur un même* `Processeur` *dans au moins deux* `Version` *différentes.*

Question 1.4 (1 point) *Écrire une requête SQL qui caractérise les* `Logiciel` *compilés dans au plus 5* `Binaire` *différents.*

Question 1.5 (2 points) *Écrire une requête SQL qui caractérise les* `Fichier` *utilisés dans un* `Logiciel` *compilé dans moins de 2* `Binaire` *différents, et n'étant pas utilisés dans un* `Logiciel` *compilé dans plus de 9* `Binaire` *différents.*

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

$$\begin{aligned} R \;=\; & \pi[\texttt{Version,Logiciel,Processeur}](Codes) - \\ & \pi[\texttt{C1.Version,C1.Logiciel,C1.Processeur}](\sigma[\; \texttt{C1.Version} \neq \texttt{C2.Version} \\ & \qquad \wedge\; \texttt{C1.Logiciel} = \texttt{C2.Logiciel} \\ & \qquad \wedge\; \texttt{C1.Processeur} = \texttt{C2.Processeur} \\ & \qquad](\alpha[\texttt{Codes : C1}] \times \alpha[\texttt{Codes : C2}])) \end{aligned}$$

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

Question 1.7 (2 points) *Écrire une requête SQL qui caractérise les* `Fichier` *utilisés dans tous les* `Binaire`*.*

Les questions suivantes portent sur la normalisation de la relation `Codes`*.*

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

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

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* `Codes` *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.*

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.*

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*		$(S, \{A, B\}, \{ecriture(A)\})$
	t4.1	$dem(update(tuple))$	$(S, \{A, B\}, \{ecriture(A), ecriture(B)\})$
⋮	⋮	⋮	⋮

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* `Wait-Die` *de l'évitement. Les transactions doivent se terminer par un* `COMMIT` *après leur update réussi.*

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

Exercice 3 (Transactions (3 points))

Question 3.1 (1 point) *Donner les définitions de "état cohérent" et de "état correct" dans un SGBD relationnel.*

Question 3.2 (2 points) *Donner les définitions de "unité logique de travail" et de "transaction" dans un SGBD relationnel. Expliquer pourquoi une transaction est une unité logique de travail.*