Bases de Données I (J. Wijsen)

18 janvier 2013

NOM + PRENOM :

Orientation + Année :

Cet examen contient 14 questions. Durée : 2 heures et 30 minutes.

Question 1 Soit R un nom de relation avec schéma ABC. Considérez la requête q_1 :

$$q_1 = \{x, y \mid \exists z\Big(R(x, y, z) \wedge \neg\exists z(R(z, y, x))\Big)\}$$

Donnez une requête en algèbre relationnelle qui est équivalente à q_1. Détaillez votre réponse.

. . ./5

Noter que la sous-formule $\Big(R(x, y, z) \wedge \neg\exists z(R(z, y, x))\Big)$ contient une occurrence libre de z et une occurrence liée de z. La requête q_2 ci-dessus est équivalente à q_1 :

$$q_2 = \{x, y \mid \exists z(R(x, y, z)) \wedge \neg\exists u(R(u, y, x))\}.$$

Cette requête se traduit en

$$\pi_{AB}(R) - \rho_{C\to A}(\pi_{BC}(R)).$$

Question 2 Considérez la requête :

$$\{x, y \mid \exists u \exists w(R(x, y, u) \vee R(w, x, y))\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

☐ Cette requête est *domain independent*.

☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

. . ./2

La requête est équivalente à

$$\{x, y \mid \exists u(R(x, y, u)) \vee \exists w(R(w, x, y))\}.$$

Puisque les sous-formules $\exists u(R(x, y, u))$ et $\exists w(R(w, x, y))$ ont les mêmes variables libres, la requête est *safe* et donc *domain-independent*.

Question 3 Considérez la requête :

$$\{y \mid \neg \exists x(S(x, y) \wedge \neg R(y))\}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

☐ Cette requête est *domain independent*.

☐ Cette requête n'est pas *domain independent*.

Expliquez en détail.

. . ./3

La requête est équivalente à

$$\{y \mid \neg \exists x(S(x, y)) \vee R(y)\}.$$

Soit a une constante qui n'appartient pas au domaine actif. Puisque $\neg \exists x(S(x, a))$ est Vrai, on a que $\neg \exists x(S(x, a)) \vee R(a)$ est Vrai. La réponse contient donc chaque constante qui n'appartient pas au domaine actif. En conséquence, cette requête n'est pas *domain independent*.

Question 4 Détaillez le protocole Wait-Die et argumentez pourquoi ce protocole évite les verrous mortels.

. . ./5

Pour gérer la concurrence, le protocole 2PL introduit des verrous. Si une transaction T_i fait une demande $S_i(A)$ ou $X_i(A)$, elle risque d'être mise "en attente", ce qui peut engendrer des verrous mortels. Un verrou mortel s'est produit si le WAIT-FOR graphe contient un cycle.
Supposons que l'indice (ou l'estampille) i dans T_i dénote le temps de démarrage de la transaction. On peut donc dire que T_i est plus âgée que T_j si $i < j$. L'objectif du protocole Wait-Die est de garantir que pour toute arête $T_i \rightarrow T_j$ dans le WAIT-FOR graphe, T_i est plus âgée que T_j, i.e., $i < j$. Cela évitera les verrous mortels, car au long de chaque chemin dans le WAIT-FOR graphe, les indices ne feront qu'augmenter. Un chemin qui part de T_i ne peut donc pas revenir en T_i.
Le protocole Wait-Die stipule alors qu'une transaction T_i est annulée si elle fait une demande de verrou qui conduirait à une situation où T_i devrait attendre (le relâchement d'un verrou par) une transaction T_j plus âgée (i.e., $i > j$).
Si une transaction T_i annulée est redémarrée, elle gardera l'estampille i. De cette façon, elle "veillit" et est sûre d'aboutir un moment.
Le désavantage de ce protocole est le grand nombre d'annulations de transactions.

Question 5 Soit $\mathcal{A}$ l'algèbre qui contient tous les opérateurs de SPJRUD **sauf la sélection** $\sigma_{A=B}$. Noter que la sélection $\sigma_{A=\text{“}c\text{”}}$ fait partie de $\mathcal{A}$. Soit R une relation avec un seul attribut A et soit q_0 la requête SPJRUD $\sigma_{A=B}(R \bowtie \rho_{A\to B}R)$. Prouvez que l'algèbre $\mathcal{A}$ ne contient aucune requête qui est équivalente à q_0.

Suggestion : Il semble vrai que pour chaque requête q_1 en $\mathcal{A}$ qui rend une relation avec schéma AB, il existe deux constantes c, d ($c \neq d$) telles que pour la relation $R = \{\{A : c\}, \{A : d\}\}$, on a que $\{A : c, B : c\} \in q_1(R)$ implique $\{A : c, B : d\} \in q_1(R)$.

. . ./5

On regarde d'abord si ce qu'il "semble vrai" est effectivement vrai. . .
Soit q une requête quelconque dans $\mathcal{A}$. Il suffit de prouver que q n'est pas équivalente à $\sigma_{A=B}(R \bowtie \rho_{A\to B}R)$. Puisque q est de taille finie, on peut toujours trouver deux constantes qui n'apparaissent pas dans q. Soient 0, 1 deux constantes qui n'apparaissent pas dans q. Disons qu'une relation avec n atrributs est *complète* si elle contient toutes les 2^n combinaisons de 0 et 1. Les relations suivantes sont donc complètes.

A_1
0
1

A_1	A_2
0	0
0	1
1	0
1	1

A_1	A_2	A_3
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

. . .

Soit R la relation

A
0
1

. On démontre par induction sur la syntaxe de q que la relation $q(R)$ (i.e., le résultat d'appliquer q sur R) est soit vide soit complète. C'est évidemment le cas pour la base de l'induction où q est simplement R, i.e., $q(R) = R$. Autrement, q est d'une des six formes ci-dessus. L'hypothèse d'induction est que chacune des relations $q_0(R)$ et $q_1(R)$ est soit vide soit complète.

Cas où q est de la forme $\sigma_{A=\text{“}c\text{”}}(q_0)$. Puisque 0, 1 n'apparaissent pas dans q, on sait $0 \neq c \neq 1$. Évidemment, $q(R)$ est vide.

Cas où q est de la forme $\pi_X(q_0)$. Il est facile à vérifier que la projection d'une relation complète est complète. Il est trivial que la projection d'une relation vide est vide. Puisque la relation $q_0(R)$ est vide ou complète par l'hypothèse d'induction, on aura que $q(R)$ est vide ou complète.

Cas où q est de la forme $\rho_{A\to B}(q_0)$. Ce cas est facile à traiter.

Cas où q est de la forme $q_0 \bowtie q_1$. Il est facile à vérifier que la jointure de deux relations complètes rend une relation complète. Il est trivial qu'une jointure est vide si un des deux arguments est une relation vide. Puisque chacune des relations $q_0(R)$ et $q_1(R)$ est vide ou complète par l'hypothèse d'induction, on aura que $q(R)$ est vide ou complète.

Cas où q est de la forme $q_0 \cup q_1$. Ce cas est facile à traiter : $q(R)$ sera vide si $q_0(R)$ et $q_1(R)$ sont toutes les deux vides ; autrement $q(R)$ sera complète.

Cas où q est de la forme $q_0 - q_1$. Ce cas est facile à traiter : $q(R)$ sera complète si $q_0(R)$ est complète et $q_1(R)$ est vide ; autrement $q(R)$ sera vide.

On peut donc conclure que la relation $q(R)$ est soit vide soit complète.

L'expression $\sigma_{A=B}(R \bowtie \rho_{A\to B}R)$ rend la relation

A	B
0	0
1	1

, qui est ni vide ni complète. Cette relation n'est donc pas équivalente à q.

Question 6 Considérez le schéma avec attributs $ABCDEF$ et les DF suivantes :

$$\begin{array}{llll} CD \to B & EF \to A & ABE \to FC & AE \to F \\ A \to E & C \to D & AB \to C & \end{array}$$

Cochez chaque case (éventuellement plusieurs) qui précède une expression correcte :

- ☐ Ce schéma est en BCNF.
- ☐ Ce schéma n'est pas en BCNF.
- ☐ Ce schéma est en 3NF.
- ☐ Ce schéma n'est pas en 3NF.

Détaillez les calculs qui mènent à cette conclusion.

.../10

D'abord, on peut simplifier l'ensemble des DF comme suit :

$$\begin{array}{llll} C \to B & EF \to A & & A \to F \\ A \to E & C \to D & AB \to C & \end{array}$$

Il y a deux façons de démontrer que D ne fait partie d'aucune clé.

Démonstration par raisonnement. Supposons par contradiction que K est une clé qui contient D. On sait que K détermine tous les attributs. Cependant, puisque D n'apparaît pas à gauche d'une flêche, il faut que $K \setminus \{D\}$ détermine tous les attributs, ce qui contredit le fait que K est une clé.

Démonstration par calcul brut. Par calcul brut, on trouve que l'ensemble des clés est $\{AB, AC, BEF, CEF\}$.

C détermine BCD. Pour la DF $C \to D$, on a que C ne détermine pas tous les attributs et D ne fait partie d'aucune clé. Le schéma n'est donc pas en 3NF.
Puisque chaque schéma en BCNF est en 3NF, le schéma n'est pas en BCNF.

Question 7 Soit U un ensemble d'attributs et Σ un ensemble de dépendances fonctionnelles sur U. Argumentez qu'il existe une décomposition de (U, Σ) en BCNF qui préserve le contenu.

. . ./5

Soit $\Sigma^+ = \{X \to A \mid X \subseteq U \text{ et } A \in U \setminus X \text{ et } \Sigma \models X \to A\}$, l'ensemble de toutes les DF singulières qui sont une conséquence logique de Σ.
Supposons que (U, Σ) n'est pas en BCNF. Alors, Σ^+ contient une DF $Y \to B$ telle que $\Sigma \not\models Y \to U$. On décomposera (U, Σ) en deux schémas (U_1, Σ_1) et (U_2, Σ_2) où $U_1 = YB$, $U_2 = U \setminus \{B\}$, $\Sigma_1 = \{X \to A \in \Sigma^+ \mid XA \subseteq U_1\}$ et $\Sigma_2 = \{X \to A \in \Sigma^+ \mid XA \subseteq U_2\}$. Le théorème de Heath nous garantit que cette décomposition est sans perte de contenu, i.e., pour chaque relation R sur U qui satisfait Σ, on a $R = \pi_{U_1}(R) \bowtie \pi_{U_2}(R)$.
Il est évident que $Y \to B \in \Sigma_1$ et $Y \to B \notin \Sigma_2$. Le point essentiel est de remarquer que la DF $Y \to B$ ne cause plus de violation de BCNF, parce que $\Sigma_1 \models Y \to U_1$.
Si (U_1, Σ_1) ou (U_2, Σ_2) n'est pas en BCNF, on applique une même décomposition. On continue à décomposer jusqu'au moment où chaque composant est en BCNF. Cette procédure se terminera, car Σ^+ est fini et chaque décomposition enlève une DF qui cause une violation de BCNF.
Supposons que la procédure se terminera avec les schémas (U'_1, Σ'_1), (U'_2, Σ'_2), …, (U'_n, Σ'_n), chacun en BCNF. Puisque chaque décomposition était sans perte de contenu, il sera vrai (pourquoi exactement ?) que pour chaque relation R sur U qui satisfait Σ, on aura $R = \pi_{U'_1}(R) \bowtie \pi_{U'_2}(R) \ldots \bowtie \pi_{U'_n}(R)$.

Question 8 Le *Tour des Flandres* et *Liège-Bastogne-Liège* sont deux courses cyclistes organisées annuellement au printemps. La table suivante sert à stocker les résultats de ces deux courses. La première ligne, par exemple, indique que le 1er avril 2012, Philippe Gilbert a terminé 75ème dans le Tour des Flandres. Il a disputé cette course avec le dossard numéro 11. Un coureur peut changer d'équipe d'une année à l'autre, mais pas pendant les quelques semaines qui séparent le *Tour des Flandres* et *Liège-Bastogne-Liège*. Par exemple, Philippe Gilbert faisait partie de l'équipe OMEGA en 2011, et de BMC en 2012. Quelles sont les dépendances fonctionnelles (DF) que l'on peut raisonablement imposer sur cette table ? Pour chaque DF, **exprimez la signification en français**. Par example,

`Coureur, Année` $\rightarrow$ `Équipe` signifie qu'un coureur ne change pas d'équipe au cours d'une même année.

Course	Coureur	Équipe	Année	Mois	Jour	Classement	Dossard
Tour des Flandres	GILBERT Philippe	BMC	2012	avril	1	75	11
Liège-Bastogne-Liège	GILBERT Philippe	BMC	2012	avril	22	16	1
Liège-Bastogne-Liège	IGLINSKIY Maxim	ASTANA	2012	avril	22	1	25
Liège-Bastogne-Liège	GILBERT Philippe	OMEGA	2011	avril	24	1	101
Tour des Flandres	GILBERT Philippe	OMEGA	2010	avril	4	3	93

. . ./10

`Coureur`, `Année` $\rightarrow$ `Équipe`
`Course`, `Année`, `Dossard` $\rightarrow$ `Coureur`
`Course`, `Année`, `Coureur` $\rightarrow$ `Classement`
`Course`, `Année`, `Classement` $\rightarrow$ `Dossard`
`Course`, `Année` $\rightarrow$ `Mois`, `Jour`
`Année`, `Mois`, `Jour` $\rightarrow$ `Course`

Question 9 Considérez l'exécution suivante :

$$W_2(A)R_1(A)R_3(B)W_2(B)R_1(B)$$

Est-ce que cette exécution est possible en 2PL ? Complétez l'exécution avec des demandes de verrous ou argumentez pourquoi cette exécution n'est pas possible en 2PL.

…/5

	$X_2(A)$	
	$W_2(A)$	
	$X_2(B)$	
	$U_2(A)$	
$S_1(A)$		
$R_1(A)$		
		$R_3(B)$
	$W_2(B)$	
$R_1(B)$		

Le début (commandes en rouge) ne nous laisse peu de liberté. En tout cas, la transaction T_2 possèdera un verrou exclusif sur B au moment où T_3 doit exécuter $R_3(B)$, ce qui est impossible en 2PL.

Question 10 Cochez la case qui précède une expression correcte :

- ☐ L'exécution de la question 9 est sérialisable.
- ☐ L'exécution de la question 9 n'est pas sérialisable.

Argumentez votre réponse.

…/5

Les arêtes dans le graphe de précédence sont $T_3 \to T_2$ et $T_2 \to T_1$. Puisque le graphe ne contient pas de cycle, l'exécution est sérialisable.

Question 11 Les *Foulées montoises* est une série annuelle de plusieurs courses à pieds qui se disputent dans la région montoise. La table `COURSES` enregistre les villes, dates et distances des courses pour l'année 2012. La table `PARTICIPANTS` enregistre le nom, genre et année de naissance de chaque participant. Chaque participant est identifié par un numéro unique. La table `RESULTATS` enregistre les temps réalisés ; par exemple, la première ligne indique qu'Anne Dua a terminé le Challenge de la Citadelle en 38 minutes et 41 secondes. Il est possible qu'un coureur déclare forfait pour une ou plusieurs courses.

`PARTICIPANTS`

Numéro	Nom	Genre	Naissance
122	Anne Dua	F	1964
133	Pierre Dupont	M	1978
144	Sven Nijs	M	1978

`RESULTATS`

Date	Coureur	Temps
12/4	122	0:38:41
26/4	122	0:39:36
12/4	133	0:35:28
26/4	133	0:39:36
1/5	133	0:33:56
1/5	144	0:30:41

`COURSES`

Ville	Jour	Distance	Nom
Mons	12/4	10km	Challenge de la Citadelle
Casteau	19/4	12km	Course du Ravel
Nimy	26/4	10km	Mémorial Plisnier
Mons	1/5	20km	Les 20km de Mons

Pour ces trois tables, donnez toutes les contraintes de type PRIMARY KEY, FOREIGN KEY et UNIQUE.

. . ./5

```
PARTICIPANTS
    PRIMARY KEY(Numéro)
RESULTATS
    PRIMARY KEY(Date, Coureur)
    FOREIGN KEY(Coureur) REFERENCES PARTICPANTS
    FOREIGN KEY(Date) REFERENCES COURSES
COURSES
    PRIMARY KEY(Jour)
    UNIQUE(Nom)
```

Question 12 Écrivez une requête en **calcul relationnel** pour répondre à la question suivante :

Donnez les numéros des coureurs qui ont participé à toutes les courses.

Pour l'exemple, aucun coureur n'a participé à toutes les courses.

.../5

$$\{x \mid \exists v_1 \exists v_2 \exists v_3 (PARTICIPANTS(x, v_1, v_2, v_3)) \\ \wedge \forall y \forall v_4 \forall v_5 \forall v_6 \Big(COURSES(v_4, y, v_5, v_6) \rightarrow \exists v_7 RESULTATS(y, x, v_7) \Big) \}$$

Question 13 Écrivez une requête en **algèbre relationnelle** pour répondre à la question suivante :

Donnez le nom de chaque coureur qui n'a participé à aucune course de 10km.

Pour l'exemple, la réponse contient *Sven Nijs*.

.../5

Soit $R = \pi_{Coureur}(RESULTATS \bowtie \rho_{Jour \rightarrow Date}(\sigma_{Distance = \text{“}10km\text{”}}(COURSES)))$.
R contient $\{Coureur : x\}$ ssi x est le numéro d'un coureur ayant participé à une course de 10km.

Soit $S = \pi_{Numero}(PARTICIPANTS) - \rho_{Coureur \rightarrow Numero}(R)$.
S contient $\{Numero : x\}$ ssi x est le numéro d'un coureur n'ayant participé à aucune course de 10km.

La requête demandée est :

$$\pi_{Nom}(PARTICIPANTS \bowtie S).$$

Question 14 La table `PATIENTS` stocke les symptômes des patients. La table `MALADIES` stocke les symptômes des maladies.

PATIENTS	P	Symptome
	Anne	fièvre
	Anne	maux de tête
	Anne	perte d'appétit
	Anne	constipation
	Pierre	fièvre
	Pierre	maux de tête

MALADIES	M	Symptome
	grippe	fièvre
	grippe	maux de tête
	grippe	perte d'appétit
	appendicite	constipation
	appendicite	fièvre
	appendicite	perte d'appétit

Écrivez **une seule** requête en **SQL** pour répondre à la question suivante :

> Donnez chaque paire (p, m) telle que p est le nom d'un patient qui a tous les symptômes de la maladie m.

Pour l'exemple, la réponse est composée des paires (Anne, grippe) et (Anne, appendicite).

Explicitez la logique derrière votre requête SQL en montrant, par exemple, la requête en *tuple relational calculus* (TRC).

. . ./5

En *Tuple Relational Calculus*, en indiquant les attributs par leurs noms (au lieu de leurs positions) :

$$\{t.P, s.M \mid \; PATIENTS(t) \wedge MALADIES(s) \wedge \forall s' \in MALADIES \\ \Big(s'.M = s.M \rightarrow \exists t' \in PATIENTS(t'.P = t.P \wedge t'.Symptome = s'.Symptome)\Big)\}$$

Ou encore :

$$\{t.P, s.M \mid \; PATIENTS(t) \wedge MALADIES(s) \wedge \neg\exists s' \in MALADIES \\ \Big(s'.M = s.M \wedge \neg\exists t' \in PATIENTS(t'.P = t.P \wedge t'.Symptome = s'.Symptome)\Big)\}$$

En SQL :

```
SELECT t.P, s.M
FROM   PATIENTS AS t, MALADIES AS s
WHERE  NOT EXISTS ( SELECT *
                    FROM   MALADIES AS sprime
                    WHERE  sprime.M = s.M
                    AND    NOT EXISTS ( SELECT *
                                        FROM   PATIENTS AS tprime
                                        WHERE  tprime.P = t.P
                                        AND    tprime.Symptome = sprime.Symptome ))
```