Bases de Données II, Mons, 29 mai 2024

NOM + PRÉNOM :	
Orientation + Année :	

Cet examen contient 9 questions. Durée : exactement 3 heures. Les questions sont censées être claires. Aucune clarification supplémentaire ne sera donnée pendant l'examen. Si une question vous semble ambiguë ou incomplète, écrivez vos hypothèses et répondez en fonction de celles-ci. **Il est permis de détacher la dernière page.**

Un fleuriste livre des bouquets à la maison. Chaque espèce de fleur (tulipe, rose…) a un prix exprimé en centimes d'euro. Par exemple, le prix d'une rose est de 150 centimes d'euro, indépendamment de sa couleur ; voir la ligne

```
<fleur fnom="rose" prix="150"/>
```

Un bouquet rassemble des fleurs de différentes espèces et couleurs. Par exemple, le bouquet *Exotique* est composé d'un seul tournesol et trois iris bleus. Voir les lignes :

```
<bouquet bnom="Exotique">
   <fleur fnom="tournesol" couleur="jaune" nombre="1"/>
   <fleur fnom="iris" couleur="bleu" nombre="3"/>
</bouquet>
```

La figure 1 montre le document XML et la figure 2 montre le DTD.

La balise `ventes-par-jour` est utilisée pour encoder les ventes journalières. Par exemple, à la date du 18 septembre 2022, le fleuriste a vendu trois bouquets *Valentin* et six bouquets *Printemps*. Voir les lignes :

```
<date mois="sep 2022" jour="18">
  <vente bnom="Valentin">3</vente>
  <vente bnom="Printemps">6</vente>
</date>
```

Question 1 Écrivez une requête en **XPath** qui renvoie le nom de chaque bouquet à l'exception de celui avec le plus petit nombre de fleurs. **La fonction `sum` est la seule fonction d'agrégation pouvant être utilisée. Notamment, l'utilisation de `max` et `min` est interdite.** Évitez l'usage des axes `parent` et `ancestor`.

Pour le document XML de la figure 1, la réponse est comme suit :

```
bnom="Valentin"
bnom="Belge"
bnom="Printemps"
```

Notez que le bouquet *Exotique*, étant le plus petit avec seulement 4 fleurs, ne fait pas partie de la réponse.

…/5

```
//bouquet[sum(fleur/@nombre)>//bouquet/sum(fleur/@nombre)]/@bnom
```

Question 2 Écrivez une requête en **XPath** qui renvoie le nom de chaque bouquet ne partageant aucune espèce de fleur avec un autre bouquet. **Il n'est pas permis d'utiliser des fonctions d'agrégation telles que `sum`, `count`, `min` et `max`.** Évitez l'usage des axes `parent` et `ancestor` dans la mesure du possible.

Pour le document XML de la figure 1, la réponse est comme suit :

```
bnom="Exotique"
```

Notez que le bouquet *Exotique* est composé d'un tournesol et des iris, deux espèces que l'on ne trouve dans aucun autre bouquet.

.../5

```
//bouquets/bouquet[not(fleur/@fnom=preceding-sibling::bouquet/fleur/@fnom)]
                  [not(fleur/@fnom=following-sibling::bouquet/fleur/@fnom)]/@bnom
```

Question 3 Pour le schéma

$$\{ABCE, AF, ACEF, ADF, BE, BC, CDE\},$$

cochez la case qui précède une assertion correcte :

☐ le schéma est α-acyclique ;

☒ le schéma n'est pas α-acyclique.

Si le schéma est α-acyclique, donnez un arbre de jointure pour ce schéma. Si le schéma n'est pas α-acyclique, expliquez pourquoi.

.../10

Si $O \subseteq T$, il est évident que O est une oreille de T. Par conséquent, l'algorithme GYO nous permet de retirer AF, BE et BC, ce qui nous laisse :

$$\{ABCE, ACEF, ADF, CDE\}.$$

Puisque $ABCE$ est maintenant une oreille de $ACEF$, l'algorithme GYO nous permet de retirer $ABCE$, ce qui nous laisse :

$$\{ACEF, ADF, CDE\}.$$

Pour chaque $O, T \in \{ACEF, ADF, CDE\}$ avec $O \neq T$, il existe un attribut dans $O \setminus T$ qui appartient au troisième schéma. Par exemple, $ACEF \setminus ADF$ contient C, qui fait partie de CDE. Par conséquent, il n'y a plus d'oreilles à retirer.

Question 4 Écrivez une requête en **XQuery** qui renvoie, pour chaque espèce de fleur vendue le 30 juin 2022, le nombre d'unités vendues à cette date. L'usage de la fonction `distinct-values` est permis.

Pour le document XML de la figure 1, la réponse est :

```
<sales>  <sale fnom="rose">50</sale>
         <sale fnom="tulipe">30</sale>
         <sale fnom="dahlia">3</sale>   </sales>
```

Par exemple, à la date du 30 juin 2022, les bouquets vendus contenant des tulipes étaient : 1 bouquet *Printemps* (avec 4 tulipes) et 2 bouquets *Belge* (avec 13 tulipes chacun), totalisant ainsi $(1 * 4) + (2 * 13) = 30$ tulipes vendues.

Formulez votre requête de manière à rendre sa logique aisément compréhensible.

. . . /10

```
let $dave := //date[@mois='juin 2022' and @jour='30']/vente
return
   <sales>{
      for $fleur in //fleurs/fleur/@fnom[.=//bouquet[@bnom=$dave/@bnom]/fleur/@fnom]
      return
         <sale fnom='{$fleur}'>{
            sum(for $b in $dave
                return $b * sum(//bouquet[@bnom=$b/@bnom]/fleur[@fnom=$fleur]/@nombre))
        }</sale>
  }</sales>
```

Question 5 Écrivez un programme en XSLT qui renvoie, pour chaque espèce de fleur (sans répétitions), le nombre total de bouquets vendus contenant cette espèce. **Il n'est pas permis d'utiliser `xsl:for-each` et `xsl:if`. Il n'est pas non plus permis d'utiliser la fonction `distinct-values`.** L'usage de la fonction `sum` est permis. Les résultats doivent être groupés comme suit :

```
<bouquets-vendus>
   <contenant fleur="tulipe">15</contenant>
   <contenant fleur="rose">16</contenant>
   <contenant fleur="iris">4</contenant>
   <contenant fleur="tournesol">4</contenant>
   <contenant fleur="dahlia">7</contenant>
</bouquets-vendus>
```

Noter, par exemple, que la tulipe fait partie des bouquets *Belge* et *Printemps*. Le nombre de bouquets *Belge* vendu est de $2+6=8$; le nombre de bouquets *Printemps* vendu est de $1+6=7$. Donc, parmi tous les bouquets vendus, il y en a 15 contenant des tulipes.

. . . /10

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<!--Pour chaque bouquet, liste le nombre d'unités vendues.-->

<xsl:template match="/">
   <bouquets-vendus>
      <xsl:apply-templates select="//fleurs/fleur"/>
   </bouquets-vendus>
</xsl:template>

<xsl:template match="fleur">
   <contenant fleur="{@fnom}">
      <xsl:value-of
       select="sum(//vente[@bnom=//bouquets/bouquet[fleur/@fnom=current()/@fnom]/@bnom])"/>
   </contenant>
</xsl:template>

</xsl:stylesheet>
```

Question 6 Simplifiez la requête (UCQ) suivante ou expliquez pourquoi aucune simplification n'est possible :

$$\begin{cases} Answer(x,y) & \leftarrow & R(x,y,z), S(z,z), R(x,y,u), S(u,w), S(w,u) \\ Answer(y,x) & \leftarrow & R(x,y,z) \end{cases}$$

Détaillez les calculs.

.../10

Soit q_1 la première requête, et q_2 la deuxième.

Soit θ la substitution $\theta \coloneqq \{x \mapsto x, y \mapsto y, z \mapsto z, u \mapsto z, w \mapsto z\}$. Soit $q_1' = \theta(q_1)$, donc

$$q_1' = Answer(x,y) \leftarrow R(x,y,z), S(z,z).$$

Il est clair que θ est un homomorphisme de q_1 vers q_1'. L'identité est un homomorphisme de q_1' vers q_1. D'après le *Homomorphism Theorem*, $q_1 \equiv q_1'$.

Il est évident qu'une requête est minimale si tous les atomes dans son *body* utilisent des noms de relation différents. Par conséquent, les requêtes q_1' et q_2 sont minimales.

Il reste à démontrer que $q_1' \not\sqsubseteq q_2$ et $q_2 \not\sqsubseteq q_1'$.

Pour la base de données $I \coloneqq \{R(a,b,c), S(c,c)\}$, on obtient $q_1'(I) = \{Answer(a,b)\}$ et $q_2(I) = \{Answer(b,a)\}$. Puisque $q_1'(I) \not\subseteq q_2(I)$, il est correct de conclure $q_1' \not\sqsubseteq q_2$. Puisque $q_2(I) \not\subseteq q_1'(I)$, il est correct de conclure $q_2 \not\sqsubseteq q_1'$.

En conclusion, voici la requête équivalente minimale (copiez ci-après le résultat de vos calculs) :

$$\begin{cases} Answer(x,y) & \leftarrow & R(x,y,z), S(z,z) \\ Answer(y,x) & \leftarrow & R(x,y,z) \end{cases}$$

Question 7 Un *graphe dirigé simple* est un couple $G = (V, E)$ avec V un ensemble fini de *sommets* et E un ensemble d'*arcs* (u, v) avec $u, v \in V$, $u \neq v$. On traite seulement des graphes sans sommets isolés, i.e., chaque sommet a au moins un arc sortant ou entrant. Chaque arc du graphe est coloré en une et une seule couleur.

Dans une base de données, un prédicat EDB R d'arité 3 est utilisé pour stocker les arcs du graphe et leur couleur. Par exemple, $R(a, c, \mathsf{red})$ signifie qu'il existe un arc dirigé de a à c avec la couleur rouge.

On définit un *chemin dirigé de* u_1 *à* u_n comme une séquence $\langle u_1, u_2, \ldots, u_n\rangle$ de sommets telle que $n \geq 2$ et pour chaque $i \in \{1, 2, \ldots, n-1\}$, (u_i, u_{i+1}) est un arc du graphe. La *longueur* de ce chemin est de $n-1$, i.e., la longueur est le nombre d'arcs parcourus. Noter que la condition $n \geq 2$ dans la définition précédente implique que la longueur d'un chemin n'est jamais zéro. Noter aussi qu'un sommet peut se répéter dans un chemin. On définit le *colorama* d'un chemin comme la séquence des couleurs de ses arcs. Formellement, le *colorama* du chemin $\langle u_1, u_2, \ldots, u_n\rangle$ est la sequence $\langle c_1, c_2, \ldots, c_{n-1}\rangle$ telle que pour chaque $i \in \{1, \ldots, n-1\}$, $R(u_i, u_{i+1}, c_i)$ est vrai. On dira qu'un chemin *utilise* la couleur c si c apparaît dans son colorama. Deux coloramas $\langle c_1, c_2, \ldots, c_k\rangle$ et $\langle d_1, d_2, \ldots, d_\ell\rangle$ sont *égaux* si $k = \ell$ et $c_i = d_i$ pour tout $i \in \{1, 2, \ldots, k\}$.

Par exemple, pour le graphe de la Figure 3, le colorama du chemin $\langle f, d, a, b, f, g, h, i, j\rangle$ est

$$\langle \mathsf{green}, \mathsf{red}, \mathsf{green}, \mathsf{blue}, \mathsf{pink}, \mathsf{red}, \mathsf{green}, \mathsf{blue}\rangle.$$

Écrivez un programme en Datalog avec $\neq$ et négation stratifiée pour le prédicat IDB binaire T tel que $T(x, y)$ est vrai si x et y sont des sommets distincts tels qu'il existe un chemin de x à y, ainsi qu'un chemin de y à x, tels que les coloramas de ces deux chemins sont égaux.

Pour le graphe de la Figure 3, les chemins $\langle b, e, f\rangle$ et $\langle f, d, b\rangle$ ont tous les deux le colorama $\langle \mathsf{green}, \mathsf{blue}\rangle$. Le programme doit donc renvoyer, entre autres, $T(b, f)$.

Avant de donner votre programme, expliquez en français la stratégie suivie par votre programme. Avant chaque règle (ou groupe de règles), expliquez en français ce que cette règle (ou ces règles) vont calculer.

.../10

```
%
% go(X,Y,U,V) holds true if there exist directed paths, one from X to Y
% and another from U to V, such that both paths have the same colorama.
%
go(X,Y,U,V) :- r(X,Y,C), r(U,V,C).
go(X,Z,U,W) :- go(X,Y,U,V), r(Y,Z,C), r(V,W,C).
t(X,Y) :- go(X,Y,Y,X), X!=Y.
```

Question 8 Écrivez un programme en Datalog avec $\neq$ et négation stratifiée pour le prédicat IDB binaire $S(x, y)$ tel que $S(x, y)$ est vrai si x et y sont des sommets respectant les deux conditions suivantes :

1. il existe un chemin de x à y ;
2. tout chemin de x à y utilise au moins 4 couleurs distinctes.

Pour le graphe de la Figure 3, le programme doit renvoyer, entre autres, $S(a, h)$. En effet, il existe un chemin de a à h, et il n'existe pas de chemin de a à h qui utilise moins de quatre couleurs.

Avant de donner votre programme, expliquez en français la stratégie suivie par votre programme. Avant chaque règle (ou groupe de règles), expliquez en français ce que cette règle (ou ces règles) vont calculer.

.../10

Les deux phrases suivantes sont équivalentes :

— Tout chemin de x à y utilise au moins 4 couleurs distinctes.

— Il n'existe pas de chemin de x à y utilisant au maximum trois couleurs.

Le prédicat `go(X,Y,C,D,E)` exprimera qu'il existe un chemin de `X` à `Y` tel que chaque arc de ce chemin est de couleur `C`, `D`, ou `E` ; un tel chemin utilise donc au maximum trois couleurs.

```
color(C) :- r(X,Y,C).
%
% go(X,Y,C,D,E) holds true if X and Y are vertices, and C, D, and E are colors such that
% there is a directed path form X to Y that uses no colors other than C, D, or E.
%
go(X,Y,C,D,E) :- r(X,Y,C), color(D), color(E).
go(X,Y,C,D,E) :- r(X,Y,D), color(C), color(E).
go(X,Y,C,D,E) :- r(X,Y,E), color(C), color(D).
go(X,Z,C,D,E) :- go(X,Y,C,D,E), r(Y,Z,C).
go(X,Z,C,D,E) :- go(X,Y,C,D,E), r(Y,Z,D).
go(X,Z,C,D,E) :- go(X,Y,C,D,E), r(Y,Z,E).
%
% nots(X,Y) holds true if X and Y are vertices such that
% there is a directed path from X to Y that uses no more than three different colors.
%
nots(X,Y) :- go(X,Y,C,D,E).
reachable(X,Y) :- r(X,Y,C).
reachable(X,Z) :- reachable(X,Y), r(Y,Z,C).
s(X,Y) :- reachable(X,Y), not nots(X,Y).
```

Le programme suivant utilise la même stratégie, mais est peut-être encore plus simple et élégant.

```
color(C) :- r(X,Y,C).
% isIn(U,V,W,X) is true if X is a member of {V,W,X}.
isIn(A,A,B,C) :- color(A), color(B), color(C).
isIn(B,A,B,C) :- color(A), color(B), color(C).
isIn(C,A,B,C) :- color(A), color(B), color(C).
%
go(X,Y,C,D,E) :- r(X,Y,B), isIn(B,C,D,E).
go(X,Z,C,D,E) :- go(X,Y,C,D,E), r(Y,Z,B), isIn(B,C,D,E).
%
nots(X,Y) :- go(X,Y,C,D,E).
reachable(X,Y) :- r(X,Y,C).
reachable(X,Z) :- reachable(X,Y), r(Y,Z,C).
s(X,Y) :- reachable(X,Y), not nots(X,Y).
```

Question 9 Pour le schéma $ABCDEF$, déterminez si $\bowtie [ABC, BCD, CDE, DEF]$ est une conséquence logique de l'ensemble $\{AB \to F, AC \to E, BC \to A, CD \to A\}$. Détaillez les calculs. En plus, cochez la case qui précède une assertion correcte :

☐ $\{AB \to F, AC \to E, BC \to A, CD \to A\} \models \bowtie [ABC, BCD, CDE, DEF]$;
☒ $\{AB \to F, AC \to E, BC \to A, CD \to A\} \not\models \bowtie [ABC, BCD, CDE, DEF]$.

. . . /10

La relation suivante ne satisfait pas $\bowtie [ABC, BCD, CDE, DEF]$:

A	B	C	D	E	F
a	b	c	d_1	e_1	f_1
a_2	b	c	d	e_2	f_2
a_3	b_3	c	d	e	f_3
a_4	b_4	c_4	d	e	f

.

Pour satisfaire $BC \to A$, il faut que $a_2 = a$. Pour satisfaire $CD \to A$, il faut que $a_2 = a_3$. L'algorithme *The Chase* remplacera donc chaque occurrence de a_2 ou a_3 par a, comme suit :

A	B	C	D	E	F
a	b	c	d_1	e_1	f_1
a	b	c	d	e_2	f_2
a	b_3	c	d	e	f_3
a_4	b_4	c_4	d	e	f

.

Pour satisfaire $AB \to F$, il faut que $f_1 = f_2$. Pour satisfaire $AC \to E$, il faut que $e_1 = e_2 = e$. L'algorithme *The Chase* remplacera donc chaque occurrence de f_2 par f_1, et chaque occurrence de e_1 ou e_2 par e, comme suit :

A	B	C	D	E	F
a	b	c	d_1	e	f_1
a	b	c	d	e	f_1
a	b_3	c	d	e	f_3
a_4	b_4	c_4	d	e	f

.

Cette dernière relation satisfait toutes les dépendances fonctionnelles, mais ne satisfait toujours pas la dépendance de jointure $\bowtie [ABC, BCD, CDE, DEF]$. Il est donc correct de conclure $\{AB \to F, AC \to E, BC \to A, CD \to A\} \not\models \bowtie [ABC, BCD, CDE, DEF]$.

On peut vérifier que la relation suivante démontre aussi que $\{AB \to F, AC \to E, BC \to A, CD \to A\} \not\models \bowtie [ABC, BCD, CDE, DEF]$:

A	B	C	D	E	F
a	b	c	d	e	f_1
a_4	b_4	c_4	d	e	f

.

```xml
<?xml version="1.0" encoding="utf-8"?>
<fleuriste>
  <fleurs>
    <!-- Les prix sont en centimes d'euro -->
    <fleur fnom="tulipe" prix="100"/>
    <fleur fnom="rose" prix="150"/>
    <fleur fnom="iris" prix="250"/>
    <fleur fnom="tournesol" prix="300"/>
    <fleur fnom="dahlia" prix="120"/>
  </fleurs>
  <bouquets>
    <bouquet bnom="Valentin">
      <fleur fnom="rose" couleur="rouge" nombre="10"/>
    </bouquet>
    <bouquet bnom="Belge">
      <fleur fnom="tulipe" couleur="noir" nombre="3"/>
      <fleur fnom="tulipe" couleur="jaune" nombre="4"/>
      <fleur fnom="tulipe" couleur="rouge" nombre="6"/>
    </bouquet>
    <bouquet bnom="Exotique">
      <fleur fnom="tournesol" couleur="jaune" nombre="1"/>
      <fleur fnom="iris" couleur="bleu" nombre="3"/>
    </bouquet>
    <bouquet bnom="Printemps">
      <fleur fnom="rose" couleur="jaune" nombre="10"/>
      <fleur fnom="tulipe" couleur="jaune" nombre="4"/>
      <fleur fnom="dahlia" couleur="rose" nombre="3"/>
    </bouquet>
  </bouquets>
  <ventes-par-jour>
    <date mois="sep 2021" jour="14">
      <vente bnom="Exotique">4</vente>
    </date>
    <date mois="juin 2022" jour="30">
      <vente bnom="Valentin">4</vente>
      <vente bnom="Printemps">1</vente>
      <vente bnom="Belge">2</vente>
    </date>
    <date mois="sep 2022" jour="18">
      <vente bnom="Valentin">3</vente>
      <vente bnom="Printemps">6</vente>
    </date>
    <date mois="sep 2022" jour="19">
      <vente bnom="Valentin">2</vente>
      <vente bnom="Belge">6</vente>
    </date>
  </ventes-par-jour>
</fleuriste>
```

FIGURE 1 – Document XML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE fleuriste [
<!ELEMENT fleuriste (fleurs, bouquets, ventes-par-jour)>
<!ELEMENT fleurs (fleur)*>
<!ELEMENT bouquets (bouquet)*>
<!ELEMENT bouquet (fleur)*>
<!ELEMENT fleur (#PCDATA)>
<!ELEMENT ventes-par-jour (date)*>
<!ELEMENT date (vente)*>
<!ELEMENT vente (#PCDATA)>
<!ATTLIST fleur fnom CDATA #REQUIRED>
<!ATTLIST fleur prix CDATA #IMPLIED>
<!ATTLIST fleur couleur CDATA #IMPLIED>
<!ATTLIST fleur nombre CDATA #IMPLIED>
<!ATTLIST bouquet bnom CDATA #REQUIRED>
<!ATTLIST date mois CDATA #REQUIRED>
<!ATTLIST date jour CDATA #REQUIRED>
<!ATTLIST vente bnom CDATA #REQUIRED>
]>
```

FIGURE 2 – DTD

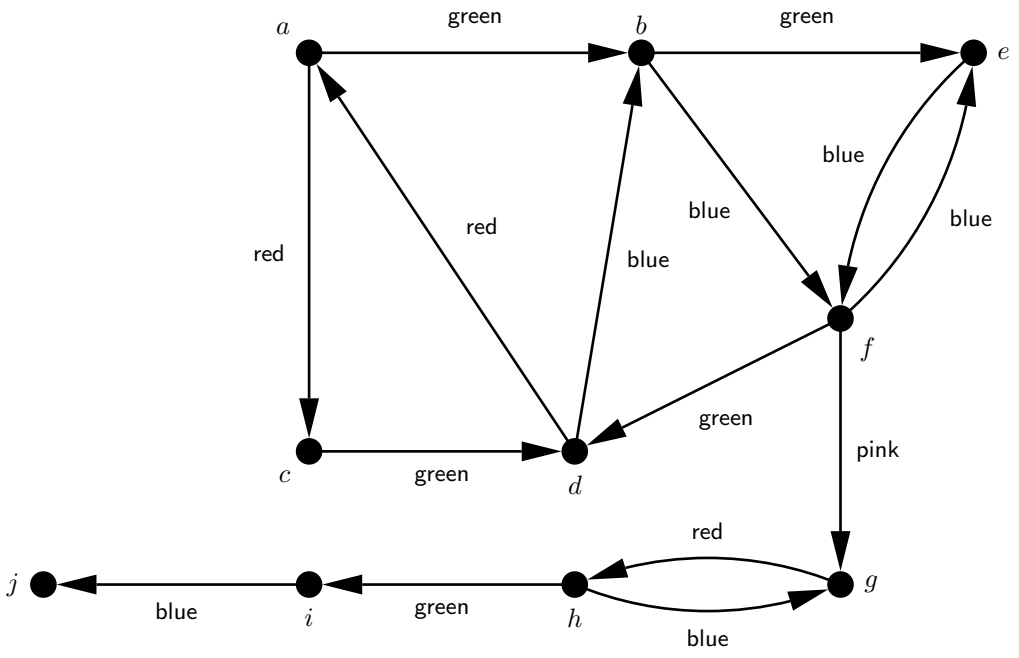

FIGURE 3 – Graphe dirigé avec des arcs colorés