Bases de Données II, Mons, 5 juin 2026

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 2 montre le document XML et son DTD.

Question 1 Rédigez un programme en XSLT qui, pour chaque couleur c unique du document, liste chaque fleur f de cette couleur, suivie des bouquets qui contiennent f dans la couleur c. Les résultats doivent être groupés comme illustré dans la figure 1, en utilisant exactement les mêmes balises et attributs. Cependant, l'ordre d'apparition des couleurs n'a pas d'importance. Pour économiser de l'espace, la figure 1 affiche le résultat en trois colonnes. Évidemment, votre programme n'affichera pas les résultats en colonnes. **Votre programme doit se limiter aux instructions XSLT enseignées au cours. Notamment, 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` ou l'instruction `<xsl:with-param>`.**

```
<CouleurFleurBouquet>
<jaune>
   <tulipe>
      <Belge />
      <Printemps />
   </tulipe>
   <tournesol>
      <Exotique />
   </tournesol>
   <rose>
     <Printemps />
   </rose>
</jaune>
```

```
<rouge>
   <rose>
      <Valentin />
   </rose>
   <tulipe>
      <Belge />
   </tulipe>
</rouge>

<blanc>
   <rose>
     <Valentin />
   </rose>
</blanc>
```

```
<noir>
   <tulipe>
      <Belge />
   </tulipe>
</noir>

<bleu>
   <iris>
      <Exotique />
   </iris>
</bleu>

<rose>
   <dahlia>
      <Printemps />
   </dahlia>
</rose>
</CouleurFleurBouquet>
```

Figure 1 – Le résultat du programme XSLT de la question 1.

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
<CouleurFleurBouquet>
   <xsl:apply-templates select="//@couleur[not(.=following::*/@couleur)]"/>
</CouleurFleurBouquet>
</xsl:template>

<xsl:template match="@couleur">
<xsl:element name="{.}">
   <xsl:apply-templates select="//bouquet/fleur
                                    [@couleur=current()]
                                    [not(@fnom = following::fleur[@couleur=current()]/@fnom)]"/>
</xsl:element>
</xsl:template>

<xsl:template match="fleur">
<xsl:element name="{@fnom}">
   <xsl:apply-templates
      select="//bouquet[fleur[@fnom=current()/@fnom][@couleur=current()/@couleur]]"/>
</xsl:element>
</xsl:template>

<xsl:template match="bouquet">
   <xsl:element name="{@bnom}"/>
</xsl:template>

</xsl:stylesheet>
```

Question 2 Rédigez une requête en **XQuery** pour résoudre le même problème que celui décrit à la question 1, en veillant à formater le résultat de manière identique.

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

.../10

```
<CouleurFleurBouquet>{
for $c in distinct-values(//@couleur)
return
   element {$c}{
      for $f in distinct-values(//bouquet/fleur[@couleur=$c]/@fnom)
      return
         element {$f}{
             for $b in //bouquet[fleur[@couleur=$c and @fnom=$f]]/@bnom
             return
                element {$b} {}
  }
 }
}</CouleurFleurBouquet>
```

Question 3 Rédigez une requête **XPath** qui renvoie le nom de chaque bouquet contenant au moins trois couleurs distinctes et comportant au moins une fleur rouge. **L'utilisation des fonctions d'agrégation (`count`, `sum`, ...) et de la fonction `distinct-values` est interdite.** Évitez également l'utilisation des axes `parent` et `ancestor`.

Pour le document XML de la figure 2, la réponse est la suivante :

```
bnom="Belge"
```

En effet, le bouquet `Belge` contient trois couleurs, dont le rouge. **Assurez-vous que les crochets et les parenthèses sont correctement imbriqués.**

.../5

```
//bouquet[fleur/@couleur="rouge"
          and
          fleur[@couleur!="rouge"]/@couleur != fleur[@couleur!="rouge"]/@couleur]/@bnom
```

Question 4 Rédigez une requête **XPath** qui renvoie le nom de chaque bouquet dans lequel aucune espèce de fleur n'existe en plusieurs couleurs différentes. **L'utilisation des fonctions d'agrégation (`count`, `sum`, ...) et de la fonction `distinct-values` est interdite.** Évitez également l'utilisation des axes `parent` et `ancestor`.

Pour le document XML de la figure 2, la réponse est la suivante :

```
bnom="Exotique"
bnom="Printemps"
```

Notez que le bouquet `Belge` n'apparaît pas dans le résultat, car il contient des tulipes de différentes couleurs, et que le bouquet `Valentin` n'y figure pas non plus, car il contient des roses de différentes couleurs. **Assurez-vous que les crochets et les parenthèses sont correctement imbriqués.**

.../5

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

Question 5 Pour le schéma

$$\{AB, BC, CD\},$$

considérez le programme *semijoin* suivant :

$$BC \coloneqq BC \ltimes AB$$
$$CD \coloneqq CD \ltimes BC$$
$$AB \coloneqq AB \ltimes BC$$
$$BC \coloneqq BC \ltimes CD$$

Est-ce que ce programme est un *full reducer* ?

Cochez la case qui précède la réponse correcte :

☐ oui ;

☒ non.

Justifiez votre réponse.

…/10

Considérons la base de données suivante, où chaque tuple est dangling :

A	B
a	b

B	C
b	c

C	D

Voici le résultat de l'application du programme semijoin :

A	B
a	b

B	C

C	D

Il reste un tuple.

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

$$\begin{cases} (q_1) & S(x,v) \coloneqq R(x,y),\ R(y,z),\ R(z,x),\ R(z,v). \\ (q_2) & S(x,v) \coloneqq R(x,z),\ R(z,y),\ R(y,x),\ R(z,v). \\ (q_3) & S(x,v) \coloneqq R(x,y),\ R(y,z),\ R(z,x),\ R(y,v),\ R(z,v). \end{cases}$$

Détaillez les calculs.

.../10

$q_3 \sqsubseteq q_1$ car l'identité est un homomorphisme de q_1 vers q_3. Donc q_3 est redondante et peut être supprimée.

Chaque homomorphisme θ de q_1 vers q_2 doit respecter $S(\theta(x), \theta(v)) = S(x, v)$, donc $\theta(x) = x$ et $\theta(v) = v$. L'atome $R(z, v)$ de q_1 impose que $R(\theta(z), v)$ soit un atome de q_2. Dans q_2, le seul atome ayant v en deuxième position est $R(z, v)$, donc $\theta(z) = z$. Considérons maintenant l'atome $R(z, x)$ de q_1. Il doit être envoyé sur un atome $R(\theta(z), \theta(x)) = R(z, x)$ dans q_2, ce qui n'existe pas. Contradiction. Donc il n'existe pas d'homomorphisme de q_1 vers q_2, et ainsi $q_2 \not\sqsubseteq q_1$.

$q_1 \not\sqsubseteq q_2$ est par un raisonnement similaire.

q_1 est minimale : chaque homomorphisme θ de q_1 vers q_1 doit respecter $\theta(x) = x$ et $\theta(v) = v$, comme x et v sont les variables libres. L'atome $R(x, y)$ impose que $R(x, \theta(y))$ doit apparaître dans q_1, ce qui laisse $\theta(y) = y$ comme seule possibilité. L'atome $R(y, z)$ impose alors que $R(y, \theta(z))$ doit apparaître dans q_1, ce qui laisse $\theta(z) = z$ comme seule possibilité. Donc, le seul homomorphisme de q_1 vers q_1 est l'identité.

q_2 est minimale par un raisonnement similaire.

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

$$\begin{cases} (q_1) & S(x,v) \coloneqq R(x,y),\ R(y,z),\ R(z,x),\ R(z,v). \\ (q_2) & S(x,v) \coloneqq R(x,z),\ R(z,y),\ R(y,x),\ R(z,v). \end{cases}$$

Quelques rappels en théorie des graphes. 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 colorié 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. Les questions 7 et 8 portent sur une telle relation ternaire R, dont un exemple est représenté dans la figure 3.

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.

Question 7 Soit $G = (V, E)$ un graphe dirigé simple. On appelle *arc de coupure* un arc $(u, v) \in E$ tel que, dans le graphe

$$G' \stackrel{\text{def}}{=} (V, E \setminus \{(u, v)\}),$$

il n'existe plus de chemin dirigé de u à v. Autrement dit, si l'on supprime (u, v), on ne peut plus atteindre v à partir de u. Écrivez un programme en **datalog**$^\neg$ pour le prédicat IDB binaire coupureGreen tel que coupureGreen(u, v) est vrai si (u, v) est un arc de coupure de couleur green. Pour le graphe de la figure 3, le programme doit renvoyer coupureGreen(c, d), coupureGreen(f, d), et coupureGreen(h, i). Notez que l'arc (a, b), de couleur green, n'est pas dans la réponse car il ne s'agit pas d'un arc de coupure, à cause du chemin $\langle a, c, d, b \rangle$.

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

```
% sans(X,Y,U,V) est vrai s'il existe un chemin
% de X à Y ne contenant pas l'arc (U,V).

    sans(X,Y,U,V) :- r(X,Y,C), r(U,V,D), X != U.
    sans(X,Y,U,V) :- r(X,Y,C), r(U,V,D), Y != V.
    sans(X,Z,U,V) :- sans(X,Y,U,V), r(Y,Z,C), Y != U.
    sans(X,Z,U,V) :- sans(X,Y,U,V), r(Y,Z,C), Z != V.
coupureGreen(U,V) :- r(U,V,green), not sans(U,V,U,V).
```

Voici une autre solution, plus élégante et plus courte :

```
% sans(X,Y,Z) est vrai s'il existe un chemin de X à Y
% n'utilisant pas l'arête (X,Z).

      sans(X,Y,Z) :- r(X,Y,C), r(X,Z,D), Y != Z.
      sans(X,Y,Z) :- sans(X,U,Z), r(U,Y,D), U != X.
      sans(X,Y,Z) :- sans(X,U,Z), r(U,Y,D), Y != Z.
coupureGreen(X,Y) :- r(X,Y,green), not sans(X,Y,Y).
```

Question 8 Soit $G = (V, E)$ un graphe dirigé simple, dont les arcs sont coloriés et encodés dans une relation ternaire R. Soient c et d deux couleurs distinctes. La *restriction* de G aux couleurs c et d est le graphe qui contient exactement les arcs de couleur c et ceux de couleur d. Écrivez un programme en **datalog**$^\neg$ pour le prédicat IDB binaire transitif, tel que transitif(c, d) est vrai si c et d sont deux couleurs distinctes et si la restriction de G aux couleurs c et d est transitive. Autrement dit, pour cette restriction, si (x, y) et (y, z) sont des arcs, alors (x, z) doit également être un arc de la restriction.

Pour le graphe de la figure 3, la réponse à cette requête est vide. Cependant, si l'on ajoutait $R(g, j, \text{brown})$ et $R(f, j, \text{pink})$, alors transitif(brown, pink) et transitif(pink, brown) seraient vrais. En effet, pour ce nouveau graphe, l'ensemble des arcs de couleur brown et pink est $\{(f, g), (g, j), (f, j)\}$, lequel est transitif.

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

```
      colors(C,D) :- r(X,Y,C), r(U,V,D), C != D.
restrict(X,Y,C,D) :- r(X,Y,C), colors(C,D).
restrict(X,Y,C,D) :- r(X,Y,D), colors(C,D).
    nottrans(C,D) :- restrict(X,Y,C,D), restrict(Y,Z,C,D), not restrict(X,Z,C,D).
       trans(C,D) :- colors(C,D), not nottrans(C,D).
```

Question 9 Considérez une base de données contenant les relations $R[A, B, C]$, $S[B, C, D]$, $T[C, D, E]$ et $U[D, E, F]$.

Écrivez un programme P en **datalog** définissant le prédicat ternaire answer tel que answer(b, c, d) soit vrai si et seulement si (b, c, d) est un tuple de S qui n'est pas *dangling* par rapport à la jointure $R \bowtie S \bowtie T \bowtie U$.

Votre programme doit respecter les contraintes suivantes :

- **aucune règle de P ne contient plus de trois variables dans son *body* ;**
- **aucune règle de P ne contient plus de deux atomes dans son *body* ;**
- **le graphe de dépendances des prédicats (PDG) de P est acyclique.**

Si votre programme ne respecte pas ces trois contraintes, la note attribuée à cette question sera de zéro.

.../10

Cette question est quasi identique à un exercice des travaux pratiques.

Voir mon message sur le forum du 13 mai 2026 : *« Suite aux TP de ce matin, j'ai ajouté la solution de l'exercice C.4. »*

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE fleuriste [
<!ELEMENT fleuriste (fleurs, bouquets)>
<!ELEMENT fleurs (fleur)*>
<!ELEMENT bouquets (bouquet)*>
<!ELEMENT bouquet (fleur)*>
<!ELEMENT fleur (#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>
]>

<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"/>
      <fleur fnom="rose" couleur="blanc" 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>
</fleuriste>
```

FIGURE 2 – Document XML précédé de son DTD

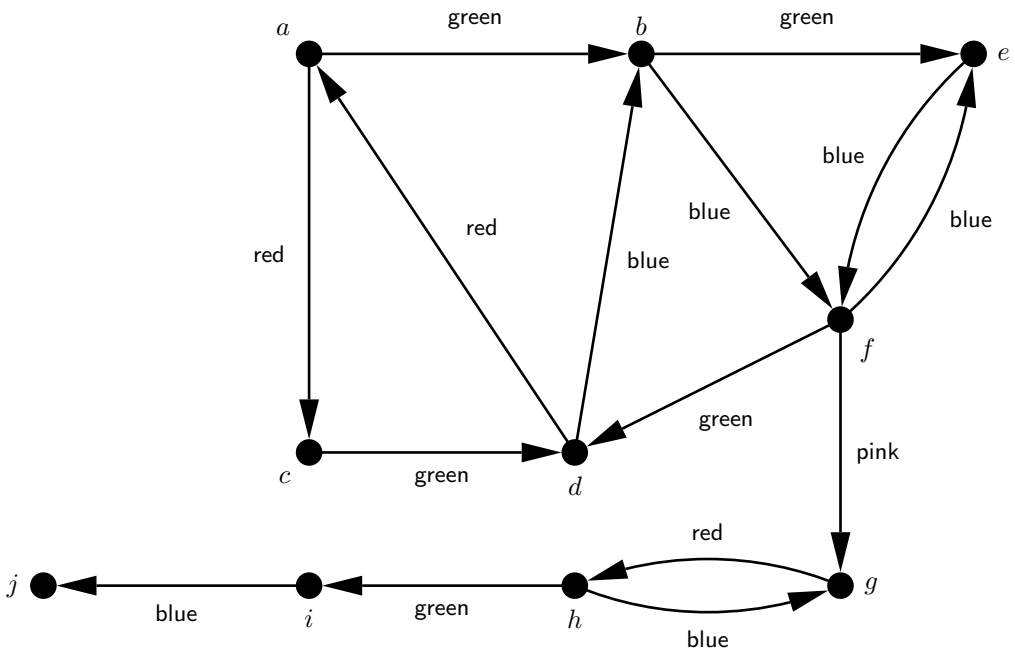

FIGURE 3 – Graphe dirigé avec des arcs coloriés