M1 Informatique – Année 2018-2019
Bases de données avancées
Épreuve écrite
J. Darmont (http://eric.univ-lyon2.fr/jdarmont/), 10/01/2019

Durée : 1h45 – Documents autorisés – Barème fourni à titre indicatif

Questions générales (2,5 points)

1. Quelles sont les différences entre le parcours d'un curseur implicite et celui d'un curseur explicite ?

2. Quelle est la caractéristique essentielle des données structurées par rapport aux données non ou semi-structurées ?

PL/pgSQL (10,5 points)

Soit la base de données dont le schéma relationnel est donné ci-dessous.

Event (eventNo, dateHeld, dateReq, dateAuth, status, estCost, estAudience)
Employee (empNo, empName, empDepartment, empPhone, empEmail)
EventPlan (planNo, eventNo#, workdate, notes, activity, empNo#))

1. Écrire une fonction de nom ratioNotApproved qui renvoie le ratio des événements (Events) dont le statut (status) n'est pas « Approved ». En cas de division par zéro (c'est le cas si la table est vide), la fonction doit renvoyer NULL.

2. Créer un type composite (enregistrement) de nom planTuple contenant les champs suivants : planNo, eventNo, workdate, notes, activity et empName. Tous ces champs sont de type chaîne de caractères, sauf workdate qui est une date.

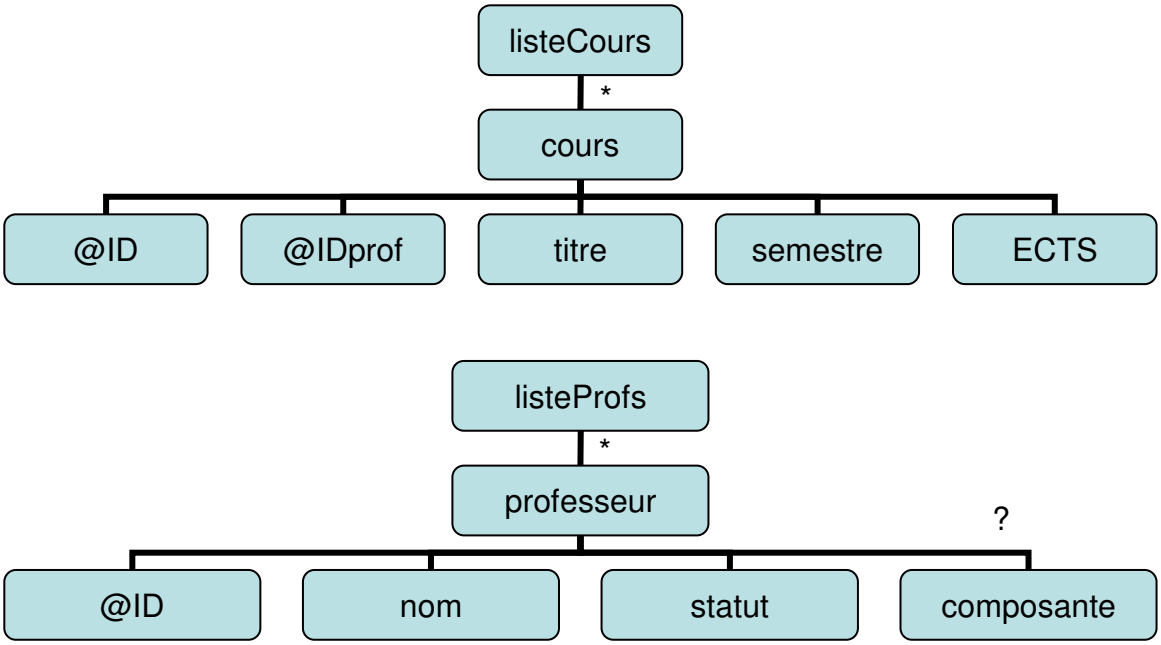
3. Écrire une fonction de nom showPlan qui renvoie toutes les caractéristiques d'une planification (EventPlan) dont le numéro (planNo) est passé en paramètre de la fonction. Le nom (empName) de l'employé (Employee) responsable doit être indiqué plutôt que son numéro (empNo). Prévoir un message d'erreur si le numéro de planification n'existe pas dans la table EventPlan : « Le numéro de planification XX n'existe pas. », où XX est le numéro de planification passé en paramètre de la fonction.

4. Écrire une fonction de nom checkDates qui renvoie les numéros des planifications pour lesquelles la date indiquée (workdate) est différente de la date (dateHeld) à laquelle s'est déroulé l'événement (Event) correspondant.

5. Pour éviter à l'avenir la vérification de la question 4, écrire un déclencheur de nom setDate sur la table EventPlan, ainsi que la fonction associée, qui affecte automatiquement à workdate la valeur de dateHeld de l'événement associé.

XQuery (7 points)

Soient les documents *cours.xml* et *profs.xml* dont les structures hiérarchiques sont représentées ci-contre. Les nœuds sont des éléments XML, à l'exception des nœuds préfixés par @, qui sont des attributs. Les * représentent des multiplicités « zéro à plusieurs » et les ? des multiplicités « zéro ou un ». Les multiplicités non précisées sont « un et un seul ».



Formuler à l'aide du langage XQuery les requêtes suivantes (utiliser, de la syntaxe XPath ou FLWOR, la plus appropriée).

1. Titres de tous les cours.

2. Caractéristiques des cours donnés par le professeur d'identifiant JD pendant le semestre 1.

3. Noms des professeurs extérieurs, c'est-à-dire qui n'appartiennent pas à une composante de l'université.

4. Titres des cours dans des éléments <s1> </s1> s'ils ont lieu au semestre 1 ou dans des éléments <s2> </s2> s'ils ont lieu au semestre 2.

5. Titres des cours et noms des professeurs correspondants, au format suivant.

```
<cours>
        <titre> </titre>
        <professeur> </professeur>
</cours>
```

6. Nombre total d'ECTS par semestre. Indiquer le numéro de semestre comme attribut dans le résultat.

7. Nombre total d'ECTS par professeur (indiquer le nom des professeurs) et par semestre. Appliquer un tri par nombre total d'ECTS décroissant.

Correction Questions générales

1. Le parcours d'un curseur implicite est plus simple à écrire que celui d'un curseur explicite, mais ne permet qu'un parcours complet du curseur, tandis qu'un curseur explicite permet tous types de parcours.

2. Les données structurées ont toujours un schéma explicite.

Correction PL/pgSQL

```
-- 1
CREATE OR REPLACE FUNCTION ratioNotApproved() RETURNS REAL AS $$
    DECLARE
        nAll INTEGER;
        nNot INTEGER;
    BEGIN
        SELECT COUNT(*) INTO nAll FROM Event;
        SELECT COUNT(*) INTO nNot FROM Event WHERE status <> 'Approved';
        RETURN nNot / nAll ::REAL;
    EXCEPTION
        WHEN division_by_zero THEN
            RETURN NULL;
    END
$$ LANGUAGE plpgsql;

-- 2
CREATE TYPE planTuple AS (
    planNo VARCHAR,
    eventNo VARCHAR,
    workdate DATE,
    notes VARCHAR,
    activity VARCHAR,
    empName VARCHAR
);

-- 3
CREATE OR REPLACE FUNCTION showPlan(no VARCHAR) RETURNS planTuple AS $$
    DECLARE
        n INTEGER;
        plan planTuple;
    BEGIN
        -- Test d'existence du numéro de planification
        SELECT COUNT(*) INTO n FROM EventPlan WHERE planNo = no;
        IF n = 0 THEN
            RAISE EXCEPTION 'Le numéro de planification % n''existe pas.', no;
        END IF;
        -- Récupération des informations de planification
        SELECT planNo, eventNo, workdate, notes, activity, empname
            INTO plan
            FROM EventPlan P, Employee E
            WHERE P.empNO = E.empNO
            AND planNo = No;
        RETURN plan;
    END
$$ LANGUAGE plpgsql;
```

```
-- 4
CREATE OR REPLACE FUNCTION checkDates() RETURNS SETOF VARCHAR AS $$
    DECLARE
        nuplet RECORD;
    BEGIN
        FOR nuplet IN SELECT planNo FROM EventPlan P, Event E
                      WHERE P.eventNo = E.EventNo AND workdate <> dateHeld
        LOOP
            RETURN NEXT nuplet.planNo;
        END LOOP;
        RETURN;
    END
$$ LANGUAGE plpgsql;

-- 5
CREATE OR REPLACE FUNCTION setDate() RETURNS TRIGGER AS $$
    BEGIN
        SELECT dateHeld INTO NEW.workdate
            FROM Event WHERE eventNo = NEW.eventNo;
        RETURN NEW;
    END
$$ LANGUAGE plpgsql;

CREATE TRIGGER setDate
BEFORE INSERT OR UPDATE ON EventPLan
FOR EACH ROW
EXECUTE PROCEDURE setDate();
```

Correction XQuery

```
(: 1 :)
//titre

(: 2 :)
//cours[@IDprof = "JD" and semestre = 1]/*

(: 3 :)
//professeur[not(composante)]/nom

(: 4 :)
for $c in //cours
return    if ($c/semestre = 1)
          then <s1>{data($c/titre)}</s1>
          else <s2>{data($c/titre)}</s2>

(: 5 :)
for  $c in //cours,
     $t in //professeur
where $c/@IDprof = $t/@ID
return    <cours>
               <titre>{data($c/titre)}</titre>
               <professeur>{data($t/nom)}</professeur>
          </cours>

(: 5bis :)
for  $c in //cours,
     $t in //professeur[@ID = $c/@IDprof]
return    <cours>
               <titre>{data($c/titre)}</titre>
               <professeur>{data($t/nom)}</professeur>
          </cours>
```

```
(: 6 :)
for $c in //cours
group by $s := $c/semestre
return <totalECTS semestre="{$s}">{sum($c/ECTS)}</totalECTS>

(: 7 :)
for  $c in //cours,
     $t in //professeur[@ID = $c/@IDprof] (: ou avec where, cf. q.5 :)
group by  $n := $t/nom,
          $s := $c/semestre
let $t := sum($c/ECTS)
order by $t descending
return <totalECTS prof="{data($n)}" semestre="{$s}">{$t}</totalECTS>
```