M1 Informatique – Année 2024-2025
Bases de données avancées
Épreuve écrite
J. Darmont (https://eric.univ-lyon2.fr/jdarmont/), 03/12/2024

Durée : 1h30 – Documents autorisés – Barème fourni à titre indicatif

PL/pgSQL (11 points)

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

Service (nos, intitulé, localisation)
Pièce (nop, désignation, couleur, poids)
Ordre (nos#, nop#, quantité)

NB : les attributs nos et nop sont de type varchar.

1. Écrire une fonction de nom chercheIntitulé() qui renvoie l'intitulé dont le numéro d'un service (nos) est passé en paramètre de la fonction. Si le numéro de service n'existe pas, lever une exception indiquant le message : « Le numéro de service S n'existe pas », où S est le numéro du service inexistant.

2. Créer un type composite (enregistrement) de nom tPièce contenant les champs suivants : nop, désignation, couleur, poids et quantité.

3. Écrire une fonction de nom ordresTous() qui renvoie toutes les pièces concernées par un ordre, avec les attributs nop, désignation, couleur, poids et quantité. Trier par numéro de pièce.

4. Créer un type composite (enregistrement) de nom tServ contenant les champs suivants : nos, intitulé et localisation.

5. Écrire une fonction de nom servicesAlt() qui renvoie les services (nos, intitulé et localisation) une fois sur deux.

6. Écrire un déclencheur de nom affectCouleur sur la table Pièce, ainsi que la fonction associée, qui affecte automatiquement à la valeur de la couleur des pièces en fonction de leur poids (cf. tableau ci-dessous).

Poids	Couleur
≤ 30	Vert
≤ 50	Orange
> 50	Rouge

TSVP

XQuery (9 points)

Soient les documents XML relatifs aux cours d'une université américaine : *courses.xml* (cours) et *instructors.xml* (professeurs), dont les arborescences d'éléments et d'attributs (ces derniers étant préfixés par @) sont représentées ci-dessous. Les caractères * et ? indiquent que l'élément ou l'attribut concerné peut apparaître plusieurs fois ou pas, respectivement, dans un même document.

```
courses
.....course*
.....@reg_num
.....subj
.....title
.....units?
.....days
.....time
.....start_time
.....end_time
.....place
.....building
.....room
```

```
instructors
.....instructor*
.....name
.....course*
```

Formuler à l'aide du langage XQuery les requêtes suivantes, en privilégiant la lisibilité du code XQuery. Tout résultat de requête doit être un fragment de code XML bien formé.

1. Titres (*title*) des cours triés par ordre alphabétique.

2. Noms (*name*) des professeurs (sans doublons).

3. Cours qui ont un coefficient (*units*).

4. Localisation (*place*) dont les noms des bâtiments (*building*) sont « PHYSIC » et « CHEM ».

5. Horaire (*time*) et localisation des jours (*days*) qui sont des mardis (code « T » comme *tuesday*).

6. Pour chaque professeur, indiquer son nom, le sujet (subj) et le titre de ses cours.

7. Nombre de cours.

8. Nombre de cours par sujets.

9. Nombre de cours par professeurs.

Correction PL/pgSQL

```
-- 1
create or replace function chercheService(no varchar) returns varchar as $$
	declare
		nbServices int;
		intit varchar;
	begin
		-- Vérification de l'existence du service
		select count(*) into nbServices from service where nos = no;
		if nbServices <= 0 then
			raise exception 'Le numéro de service % n''existe pas', no;
		end if;
		-- Récupération de l'intitulé
		select intitule into intit from service where nos = no;
		return intit;
	end
$$ language plpgsql;

select chercheService('S3');

-- 2
create type tPiece as (
	nop varchar,
	designation varchar,
	couleur varchar,
	poids numeric,
	quantite numeric
);

-- 3
create or replace function ordresTous() returns setof tPiece as $$
	declare
		nuplet tPiece;
	begin
		-- Curseur implicite, svp
		for nuplet in  select piece.nop, designation, couleur, poids, quantite
				from piece, ordre
				where piece.nop = ordre.nop
				order by piece.nop
		loop
			return next nuplet;
		end loop;
		return;
	end
$$ language plpgsql;

select ordresTous();

-- 4
create type tServ as (
	nos varchar,
	intitule varchar,
	localisation varchar
);

-- 5
create or replace function servicesAlt() returns setof tServ as $$
	declare
		cursServ cursor for   select nos, intitule, localisation
				      from service;
		nuplet tServ;
	begin
		-- Curseur explicite
		open cursServ;
		fetch cursServ into nuplet;
		while found loop
			return next nuplet;
			move cursServ;
			fetch cursServ into nuplet;
		end loop;
		close cursServ;
		return;
	end
$$ language plpgsql;

select servicesAlt();
```

```sql
-- 6
-- Fonction associée au déclencheur
create or replace function affectCouleur() returns trigger as $$
	begin
		if new.poids <= 30 then
			new.couleur := 'Vert';
		elsif new.poids <= 50 then
			new.couleur := 'Orange';
		else
			new.couleur := 'Rouge';
		end if;
		return new;
	end
$$ language plpgsql;

-- Déclencheur
create trigger affectCouleur
before insert or update
on piece
for each row
execute procedure affectCouleur();

insert into piece values('P8', 'Lavabo', null, 15);
insert into piece values('P9', 'Lavabo', null, 45);
insert into piece values('P0', 'Lavabo', null, 75);
```

Correction XQuery

(: 1 :)
```
for $t in //title
order by $t
return $t
```

(: 2 :)
```
for $n in distinct-values(
	for $i in //instructor
	return $i/name)
return <name>{$n}</name>
```

(: 3 :)
```
for $c in /courses/course[units]
return $c
```

(: 4 :)
```
for $c in /courses/course
where $c/place/building = "PHYSIC" or $c/place/building = "CHEM"
return $c/place
```

(: 5 :)
```
for $c in /courses/course
where $c/days = "T"
return <res>
		{$c/time}
		{$c/place}
	</res>
```

(: 6 :)
```
for	$i in //instructor,
	$c in /courses/course
where $c/@reg_num = $i/course
return <res>
		{$i/name}
		{$c/subj}
		{$c/title}
	</res>
```

(: 7 :)

```
let $n := count(/courses/course)
return <res>{$n}</res>
```

(: 8 :)

```
for $c in /courses/course
group by $g := $c/subj
let $n := count($c)
return <res>
                <sujet>{$g}</sujet>
                <nombre_cours>{$n}</nombre_cours>
        </res>
```

(: 9 :)

```
for     $i in //instructor,
        $c in /courses/course
where $c/@reg_num = $i/course
group by $g := $i/name
let $n := count($c)
return <res>
                <nom>{$g}</nom>
                <nombre_cours>{$n}</nombre_cours>
        </res>
```