M1 Informatique – Année 2023-2024
Bases de données avancées
Épreuve écrite
J. Darmont (https://eric.univ-lyon2.fr/jdarmont/), 20/12/2023

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

PL/pgSQL (12 points)

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

ATHLETES (ID, ATHLETE_NAME, ATHLETE_GENDER)
SPORTS (ID, SPORT)
NATIONS (ID, NATION)
OLYMPICS (ID, ATHLETE_ID#, SPORT_ID#, NATION_ID#)

1. Créer un enregistrement de nom *tAthlete* contenant les champs suivants : *nom*, *genre*, *sport* et *nation*. Tous ces champs sont de type chaîne de caractères.

2. Écrire une fonction nommée *athleteProfile()* qui renvoie le nom (ATHLETE_NAME), le genre (ATHLETE_GENDER), le sport et la nation de l'athlète dont l'identifiant numérique est passé en paramètre de la fonction. Dans le cas où l'identifiant en paramètre n'est pas présent dans la table ATHLETES, lever une exception indiquant l'identifiant erroné.

3. Créer un enregistrement de nom *tNoms* contenant les champs suivants : *nom* et *prenom*. Ces deux champs sont de type chaîne de caractères.

4. Écrire une fonction nommée *athleteNames()* qui renvoie le nom et le prénom de tous les athlètes. Les noms et prénoms (dans cet ordre) sont stockés dans l'attribut unique ATHLETE_NAME, séparés par une virgule. Afin de les dissocier, utiliser la fonction PL/pgSQL *split_part(*chaîne à couper, délimiteur, numéro de sous-chaîne*)*.

5. Écrire un déclencheur de nom *checkAthletes* et sa fonction associée, qui a deux fonctionnalités.

- Vérifier que la valeur de l'attribut ATHLETE_NAME contient bien une virgule. Dans le cas contraire, interrompre l'exécution avec le message « Nom d'athlète mal formé. »
- Convertir la valeur de l'attribut ATHLETE_GENDER en caractères minuscules.

XQuery (8 points)

Soient les documents XML relatifs aux points de vente d'une entreprise multinationale : *locations.xml* (localités), *countries.xml* (pays) et *regions.xml* (régions du monde), 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.

locations
.....location *
.....@id
.....@country_id
.....street_address
.....postal_code ?
.....city
.....state_province ?

countries
.....country *
.....@id
.....@region_id
.....name

regions
.....region *
.....@id
.....name

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. Nom (*name*) des pays (*country*) triés par ordre alphabétique.

2. Nombre de localités (*location*), de pays et de régions.

3. Villes (*city*) qui sont associées à un état ou une province (c'est-à-dire que *state_province* existe), triées par ordre

alphabétique.
4. Nom de la région d’identifiant 2.

5. Nom des villes et de leurs pays au format suivant.

```
<location>
    <city> </city>
    <country> </country>
</location>
```

6. Nom des villes et de leurs pays pour la région Americas, au même format qu’à la question 5.

7. Nombre de localités par nom de pays (trier les noms de pays par ordre alphabétique) au format suivant.

```
<group>
    <country_name> </country_name>
    <location_count> </location_count>
</group>
```

8. Nombre de localités par nom de pays et par région pour la région Europe, au même format qu’à la question 7.

Correction PL/pgSQL

```
-- 1
create type tAthlete as (
	nom varchar,
	genre varchar,
	sport varchar,
	nation varchar
);

-- 2
create or replace function athleteProfile(idAthlete numeric) returns tathlete as $$
	declare
		athlete tAthlete;
		c integer;
	begin
		-- Test d'existence de l'id athlète
		select count(*) into c from athletes where id = idAthlete;
		if c = 0 then
			raise exception 'L''identifiant d''athlète % n''existe pas.', idAthlete;
		end if;
		-- Lecture des données
		select athlete_name, athlete_gender, sport, nation into athlete
		from athletes a, sports s, nations n, olympics
		where athlete_id = a.id and sport_id = s.id and nation_id = n.id
		and a.id = idAthlete;
		-- Résultat
		return athlete;
	end
$$ language plpgsql;

-- 3
create type tNoms as (
	nom varchar,
	prenom varchar
);

-- 4
create or replace function athleteNames() returns setof tnoms as $$
	declare
		noms tNoms;
	begin
		for noms in select split_part(athlete_name, ',', 1), split_part(athlete_name, ',', 2)
		from athletes loop
			return next noms;
		end loop;
		return;
	end
$$ language plpgsql;

-- 5
create or replace function checkAthletes() returns trigger as $$
	begin
		-- Test sur la virgule
		if new.athlete_name not like '%,%' then
			raise exception 'Nom d''athlète mal formé.';
		end if;
		-- Mise en minuscule du genre
		new.athlete_gender := lower(new.athlete_gender);
		return new;
	end
$$ language plpgsql;

create trigger checkAthletes
	before insert or update
	on athletes
	for each row
	execute procedure checkAthletes();
```

Correction XQuery

(: 1 :)
for $p in //country
order by $p/name
return $p/name

```
(: 2 :)
let     $nbLoc := count(//location),
        $nbPay := count(//country),
        $nbReg := count(//region)
return <res>
                <nbLocalité>{$nbAdr}</nbLocalité>
                <nbPays>{$nbPay}</nbPays>
                <nbRegion>{$nbReg}</nbRegion>
        </res>

(: 3 :)
for $l in //location[state_province]
order by $l/city
return $l/city

(: 4 :)
for $r in //region
where $r/@id = 2
return $r/name

(: 5 :)
for     $l in //location,
        $c in //country
where $l/@country_id = $c/@id
return <location>
                {$l/city}
                <country>{data($c/name)}</country>
        </location>

(: 6 :)
for     $l in //location,
        $c in //country,
        $r in //region
where $l/@country_id = $c/@id and $c/@region_id = $r/@id and $r/name = "Americas"
return <location>
                {$l/city}
                <country>{data($c/name)}</country>
        </location>

(: 7 :)
for     $l in //location,
        $c in //country
where $l/@country_id = $c/@id
order by $c/name
group by $g := $c/name
let $n := count($l)
return <group>
                <country_name>{data($c/name)}</country_name>
                <location_count>{$n}</location_count>
        </group>

(: 8 :)
for     $l in //location,
        $c in //country,
        $r in //region
where $l/@country_id = $c/@id and $c/@region_id = $r/@id and $r/name = "Europe"
group by $gc := $c/name, $gr := $r/name
let $n := count($l)
return <group>
                <country_name>{data($c/name)}</country_name>
                <location_count>{$n}</location_count>
        </group>
```