M1 Informatique – Année 2017-2018
Bases de données avancées
Épreuve écrite
J. Darmont (http://eric.univ-lyon2.fr/~jdarmont/), 12/01/2018

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

Questions générales (2 points)

1. Citer deux manières d'exécuter des requêtes SQL dans un programme applicatif.

2. Quels sont les deux grands types de documents semi-structurés ?

PL/SQL (11 points)

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

Facility (facNo, facName)
Customer (custNo, custName, custContact, custPhone, custAddress, custCity, custState, custZip)
EventRequest (eventNo, facNo#, custNo#, dateHeld, dateReq, dateAuth, status, estCost, estAudience)

1. Écrire une procédure stockée de nom custProfile qui affiche toutes les caractéristiques du client dont le numéro (custNo) est passé en paramètre de la procédure.

2. Ajouter à la procédure custProfile un traitement d'exception au cas où le numéro de client n'existe pas dans la table Customer. Ne pas réécrire toute la procédure, indiquer par des numéros dans la réponse à la question 1 les endroits où il faut insérer du code et écrire le code correspondant à part en l'associant aux numéros.

3. Écrire un bloc PL/SQL anonyme permettant d'afficher les informations (eventNo, facName, custName, estCost, status) triées par coût estimé (estCost) décroissant.

4. Écrire un déclencheur de nom eventCheck permettant de vérifier que, pour chaque nouvelle demande (EventRequest) ou modification de demande, la date d'autorisation dateAuth est renseignée. Si ce n'est pas le cas, interrompre l'exécution avec le message « Erreur d'autorisation ». Sinon, vérifier que la date d'autorisation dateAuth est supérieure ou égale à la date de demande dateReq et que la date tenue dateHeld est supérieure ou égale à la date d'autorisation dateAuth. Si ce n'est pas le cas, interrompre l'exécution avec le message « Erreur de date ».

5. Écrire un bloc PL/SQL anonyme permettant de créer, pour chaque établissement (Facility), une vue de nom facilityNom_de_l'établissement et de schéma (eventNo, status, estCost, estAudience). Attention : les noms d'établissements (facName) peuvent contenir des espaces !

XQuery (7 points)

Soit le document *plants.xml* dont un extrait représentatif est reproduit ci-dessous.

```
<CATALOG>
     <PLANT>
          <COMMON>Bloodroot</COMMON>
          <BOTANICAL>Sanguinaria canadensis</BOTANICAL>
          <ZONE>4</ZONE>
          <LIGHT>Mostly Shady</LIGHT>
          <PRICE>$2.44</PRICE>
          <AVAILABILITY>031599</AVAILABILITY>
     </PLANT>
     (: ... :)
</CATALOG>
```

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

1. Nom commun (COMMON) de toutes les plantes (PLANT).

2. Plantes dont le code de disponibilité (AVAILABILITY) est compris entre 40000 et 60000

3. Familles (BOTANICAL) de toutes les plantes, par ordre alphabétique. Éliminer les doublons éventuels.

4. Prix (PRICE) moyen de toutes les plantes.

5. Nombre de plantes par ZONE. Trier les résultats par zone.

6. Nombre de plantes par zone et par type de lumière (LIGHT). Trier les résultats par zone et par type de lumière.

7. Noms des familles de plantes qui regroupent au moins deux plantes.

Correction Questions générales

1. SQL encapsulé dans un langage ; API ; Interface de niveau appel ; Procédure stockée.

2. Documents orientés données et documents orientés documents.

Correction PL/SQL

```
-- 1 + 2
CREATE OR REPLACE PROCEDURE custProfile(num VARCHAR) IS
  custTuple Customer%ROWTYPE;
  noData EXCEPTION;
  n INTEGER;
BEGIN
  -- Test d'existence des données
  SELECT COUNT(*) INTO n FROM Customer WHERE custNo = num;
  IF n = 0 THEN
    RAISE noData;
  END IF;
  -- Affichage des données
  SELECT * INTO custTuple FROM Customer WHERE custNo = num;
  DBMS_OUTPUT.PUT_LINE('custNo      ' || custTuple.custNo);
  DBMS_OUTPUT.PUT_LINE('custName    ' || custTuple.custName);
  DBMS_OUTPUT.PUT_LINE('custContact ' || custTuple.custContact);
  DBMS_OUTPUT.PUT_LINE('custPhone   ' || custTuple.custPhone);
  DBMS_OUTPUT.PUT_LINE('custAddress ' || custTuple.custAddress);
  DBMS_OUTPUT.PUT_LINE('custCity    ' || custTuple.custCity);
  DBMS_OUTPUT.PUT_LINE('custState   ' || custTuple.custState);
  DBMS_OUTPUT.PUT_LINE('custZip     ' || custTuple.custZip);
EXCEPTION
  WHEN NoData THEN
    RAISE_APPLICATION_ERROR(-20001, 'Invalid custNo');
END;

-- 3
DECLARE
  CURSOR eventList IS
    SELECT eventNo, facName, custName, estCost, status
    FROM EventRequest E, Facility F, Customer C
    WHERE E.facNo = F. facNo AND E.custNo = C.custNo
    ORDER BY estCost DESC;
  eventTuple eventList%ROWTYPE;
BEGIN
  FOR eventTuple IN eventList LOOP
    DBMS_OUTPUT.PUT_LINE(eventTuple.eventNo || ', ' || eventTuple.facName ||
    ', ' || eventTuple.custName || ', ' || eventTuple.estCost || ', ' ||
    eventTuple.status);
  END LOOP;
END;
```

```
-- 4
CREATE OR REPLACE TRIGGER  eventCheck
BEFORE INSERT OR UPDATE
ON EventRequest
FOR EACH ROW
DECLARE
  authError EXCEPTION;
  dateError EXCEPTION;
BEGIN
  IF :NEW.dateAuth IS NULL THEN
    RAISE authError;
  ELSE
    IF :NEW.dateAuth < :NEW.dateReq OR :NEW.dateHeld < :NEW.dateAuth THEN
      RAISE dateError;
    END IF;
  END IF;
EXCEPTION
  WHEN authError THEN
    RAISE_APPLICATION_ERROR(-20001, 'Erreur d''autorisation');
  WHEN dateError THEN
    RAISE_APPLICATION_ERROR(-20002, 'Erreur de date');
END;

-- 5
DECLARE
  CURSOR facList IS
    SELECT * FROM Facility;
  facTuple facList%ROWTYPE;
BEGIN
  FOR facTuple IN facList LOOP
    EXECUTE IMMEDIATE 'CREATE VIEW facility' ||
                      REPLACE(facTuple.facName, ' ', '') ||
                      ' AS SELECT eventNo, status, estCost, estAudience
                      FROM EventRequest WHERE facNo = ' || facTuple.facNo;
  END LOOP;
END;
```

Correction XQuery

```
(: 1 :)
//COMMON

(: 2 :)
//PLANT[AVAILABILITY >= 40000 and AVAILABILITY <= 60000]

(: 3 :)
distinct-values(  for $b in //BOTANICAL
              order by $b
              return $b )

(: 4 :)
avg(//PLANT/number(substring-after(PRICE, '$')))

(: 5 :)
for $p in //PLANT
group by $z := $p/ZONE
order by $z
return    <groupe>
          <zone>{$z}</zone>
          <nb_plantes>{count($p)}</nb_plantes>
     </groupe>
```

```
(: 6 :)
for $p in //PLANT
group by $z := $p/ZONE, $l := $p/LIGHT
order by $z, $l
return    <groupe>
          <zone>{$z}</zone>
          <lumiere>{$l}</lumiere>
          <nb_plantes>{count($p)}</nb_plantes>
     </groupe>

(: 7 :)
for $p in //PLANT
group by $b := $p/BOTANICAL
order by $b
where count($p) >= 2
return <famille>{$b}</famille>
```