M1 IDSM-Kharkiv – Année 2018-2019
Bases de données avancées
Examen
J. Darmont (http://eric.univ-lyon2.fr/~jdarmont/), 27/10/2018

Durée : 2 heures – Documents autorisés – Barème fourni à titre indicatif

Exercice 1 : Modélisation conceptuelle (5 points)

Une usine de fabrication de pièces détachées souhaite informatiser sa gestion de la fabrication, du stockage et de l'approvisionnement. Proposer un diagramme de classes UML.

Un produit semi-fini fabriqué par l'entreprise est défini par une désignation (nom), la quantité en stock et la quantité en cours de fabrication. Il est composé, en une quantité définie, de plusieurs produits de base. Un produit de base peut entrer dans la composition de plusieurs produits semi-finis. Un produit de base est défini par une désignation, la quantité en stock et la quantité en cours de commande.

Un produit de base peut être livré par plusieurs fournisseurs. Un fournisseur peut également livrer plusieurs produits. Dans chaque cas, on souhaite stocker le délai d'approvisionnement. Un fournisseur est défini par sa raison sociale (nom) et la ville d'où il livre.

L'entreprise passe des contrats avec ses fournisseurs. Un contrat est défini par une date d'effet et une durée. Il est passé avec une entreprise. Chaque entreprise peut passer plusieurs contrats.

Dans l'entreprise, les contrats sont gérés par un employé. Un employé est défini par son nom et son prénom. Il peut gérer plusieurs contrats et peut également travailler sur plusieurs contrats. Plusieurs employés peuvent travailler sur le même contrat.

Enfin, les employés travaillent dans un département de l'entreprise. Un département est défini par son nom. Il est également dirigé par un employé.

Exercice 2 : Requêtes SQL (5 points)

Le schéma relationnel de la base de données « télévision » est donné ci-dessous.

CHAINE (IdChaîne, Nom, Diffusion)
PROGRAMME (IdProg, Titre, Durée, CodeCSA, PDM, IdChaîne#, IdProd#)
ANIMATEUR (IdAnim, Nom, Prénom, Pseudo, Salaire)
STE_PRODUCTION (IdProd, RaisonSociale, Nationalité)
PRESENTER (IdProg#, IdAnim#, TpsAntenne)

Clés primaires
Clés étrangères#

Formuler en SQL les requêtes suivantes.

1. Titre des programmes dont le code CSA est strictement supérieur à 10 ou dont les parts de marchés (PDM) sont inférieures ou égales à 10.
2. Nom, prénom et pseudonyme des animateurs et noms des chaînes sur lesquelles ils présentent des programmes. Trier le résultat par ordre alphabétique de nom d'animateur et enlever les doublons.
3. Durée totale des programmes des programmes diffusés sur chaque chaîne (indiquer le nom et le mode de diffusion des chaînes).
4. Somme des ratios $\frac{PDM}{TpsAntenne}$ pour chaque programme (donner le titre des programmes). Classer les programmes du plus efficace (meilleure somme de ratios) au moins efficace
5. Nom et prénom des animateurs qui ne présentent aucun programme de durée supérieure à 15 minutes.

Exercice 3 : Programmation PL/SQL (10 points)

Le schéma relationnel de la base de données « requests » 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.

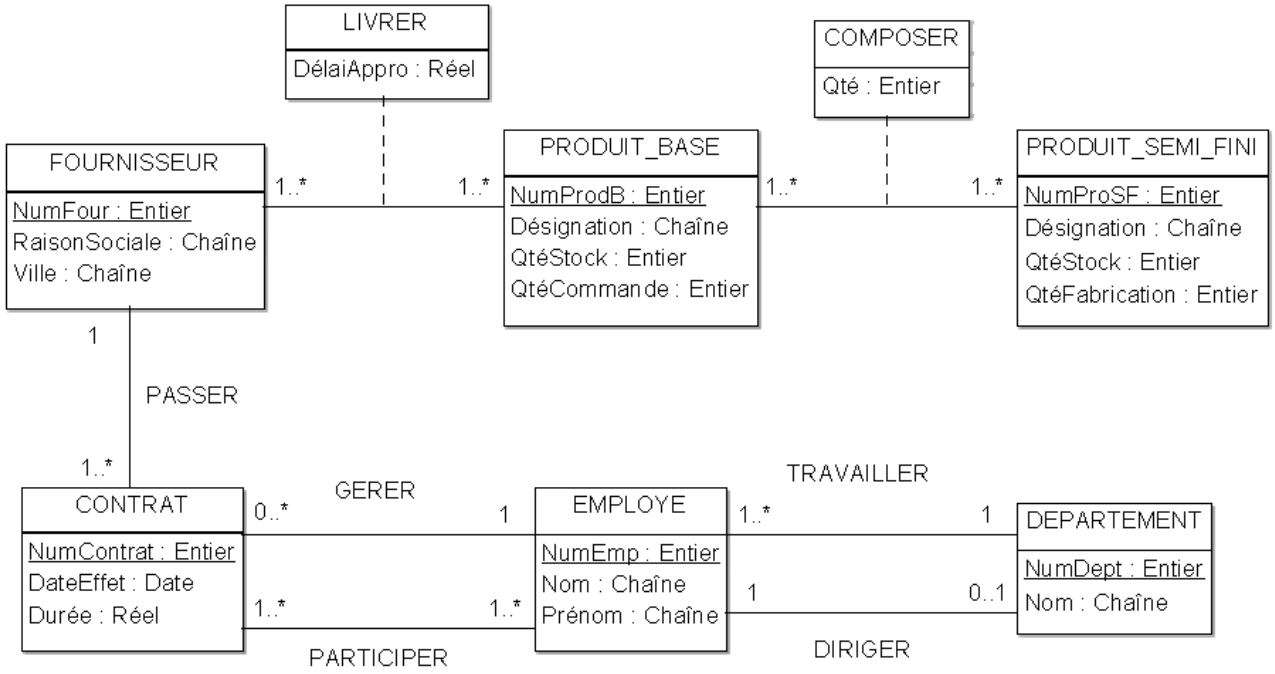
2. Ajouter à la procédure custProfile une exception si le numéro de client n'existe pas dans la table Customer.

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

4. Écrire un déclencheur de nom eventCheck qui vérifie que, pour chaque nouvelle demande (EventRequest) ou modification de demande, la date d'autorisation dateAuth existe. 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 qui crée, 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 !

Correction exercice 1



Correction exercice 2

```
--1
SELECT *
FROM PROGRAMME
WHERE CodeCSA > 10
OR PDM <= 10

--2
SELECT DISTINCT a.Nom, Prénom, Pseudo, c.Nom
FROM ANIMATEUR a, PRESENTER p, PROGRAMME g, CHAINE c
WHERE a.IdAnim = p.IdAnim
AND p.IdProg = g.IdProg
AND g.IdChaîne = c.IdChaîne
ORDER BY a.Nom

--3
SELECT Nom, Diffusion, SUM(Durée)
FROM CHAINE c, PROGRAMME p
WHERE c.IdChaîne = p.idChaîne
GROUP BY Nom, Diffusion
--4
SELECT Titre, SUM(PDM / TpsAntenne) Ratio
FROM PROGRAMME g, PRESENTER p
WHERE g.IdProg = p.IdProg
GROUP BY Titre
ORDER BY Ratio DESC

--5
SELECT Nom, Prénom
FROM ANIMATEUR
WHERE IdAnim IN (
      SELECT IdAnim from ANIMATEUR
      MINUS
      SELECT IdAnim
          FROM PRESENTER p, PROGRAMME g
          WHERE p.IdProg = g.IdProg
          AND Durée > 15)
```

Correction exercice 3

```
-- 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;
```