Projet :

La société AQUIFLOR est une entité d'un grand groupe de fleuristes dont l'activité principal couvre tout le domaine des roses. Etant donné sa récente création, elle ne dispose actuellement d'aucun système d'information lui permettant de gérer facilement ses clients, ses catalogues et ses ventes .

AQUIFLOR souhaite donc informatiser sa structure et mettre en place un série d'outils lui permettant de gérer de la manière la plus efficace possible :

- Les stocks de fleurs à l'unité
- Les stocks de compositions florales
- Les recettes de compositions florales
- Les catalogues et les tarifs
- Les clients et leurs commandes

A l'heure actuelle, un client qui vient au magasin, ne peut acheter que des fleurs à l'unité ou des compositions florales dont le contenu a été créé il y a maintenant plus de six mois et qui, pour le coup, ne sont plus au goût du jour. De plus, AQUIFLOR ne dispose pas encore de catalogue papier, ni de catalogue en ligne. Son souhait serait également de s'ouvrir sur la vente par correspondance mais tant qu'aucun système d'information n'est mis en place, elle ne peut rien envisager.

Les gérant d'AQUIFLOR espèrent donc que ce nouveau système d'information leur permettront d'être plus réactifs tant à la demande des clients qu'à l'évolution des compositions en fonction de la saison ou des roses disponibles. Ils espèrent également que ce SI leur permettra d'éditer un catalogue avec la possibilité de gérer des ventes flash et des promotions, ainsi que des opérations marketing telles que « La composition du mois » ou encore « La rose du jour ».

Enfin, et ce dans un soucis de gérance, ils souhaitent pouvoir étudier en profondeur les évolutions des ventes, les évolutions des compositions, lesquelles se vendent le plus, idem pour les roses … et tout un tas de statistiques leur permettant de mettre en place des plans de marketing pour rattraper ou améliorer les ventes.

L'objectif de votre travail sera de constituer une partie de la base de données, un référentiel des requêtes ainsi que l'écriture de certains traitements, automatisés ou non. Ce travail sera effectué en tenant compte que la base de données sera une base Oracle Express Edition 10g tournant sur un serveur Linux Ubuntu 14.04 LTS.

Documents et matériels autorisés :

- Mémento SQL
- Fiches de cours
- Calculatrice

Annexes :

- A : MCD
- B : Exemples de données
- C : Procédures et Fonctions valides
- D : Procédures et Fonctions à travailler

Règles à respecter pour le système d'information :

- Le prix de vente des roses est calculé en tenant compte d'une marge prise sur le prix d'achat.
- Le prix de vente des compositions est calculé en fonction du prix de vente des roses contenue dans le bouquet sur lequel est appliqué un pourcentage de réduction (dans le genre des pourcentages appliqués lors de commandes en gros). A l'heure actuel, ce pourcentage s'élève à 10% du prix de la rose intégrée dans une composition.
- Les groupes de composition permettent d'appliquer des réductions au « bons » clients. Un client dont le montant des commandes ne dépasse pas 500 € n'appartient à aucun groupe de remise. Un client dont le montant des commandes est compris entre 500 € et 2000 € appartient au groupe A. Un client dont le montant des commandes est compris entre 2000 € et 6000 € appartient au groupe B. Enfin, un client dont le montant des commandes dépasse 6000 € appartient au groupe C. Les affectations des clients aux groupes de remises sont effectuées par un traitement automatique quotidien exécuté tous les soirs à 22h.

Lecture et compréhension : Estimation 10 min

Bon courage …

P.S. : Une estimation du temps de réalisation est données pour chacune des parties.

Partie 1 : SQL, Requêtes simples et complexes

Travail à faire : (Estimation : 20 min)

En vous servant du MCD (Annexe A) et des exemples de données (Annexes Bx), écrire le code SQL des requêtes correspondantes au différentes attentes

Je me permet de vous rappeler que pour une requête, plusieurs solutions sont possibles.
Essayez, autant que faire se peut, d'écrire des requêtes dont le résultat est directement présentable à la direction de la société AQUIFLOR. Evitez les noms de champs à rallonge et sans signification. N'hésitez pas à utiliser des alias.

1. Ecrire le script de création des tables GROUPESREMISES, CLIENTS et COMMANDES avec les relations.
2. Lister les codes articles, les noms et descriptions des fleurs disponibles sur le catalogue par code articles croissant
3. Donner le code article et le prix de vente des roses de couleurs chaudes (jaunes et oranges) triées par prix de vente.
4. Donner le prix de vente moyen des roses de type équateurs.
5. Donner, pour chaque composition, le code article et le nombre de roses contenues dans le bouquet.
6. Lister les compositions ne contenant ni roses rouges ni roses roses ;-)
7. Ecrire la requête qui permettra de mettre à jour les prix des compositions en fonctions des fleurs les composant, en tenant compte de la réduction d'inclusion d'une rose dans une composition.

Partie 2 : SQL Avancé, PL/SQL, Procédures, Fonctions et Triggers.

Travail à faire : (Estimation : 80 min)

En vous servant des annexes fournies et des exemples de données, écrire le code PL/SQL des différentes procédures, fonctions et triggers.

1. **Question 1**

   Ecrire une fonction qui permettra de savoir si une fleur existe dans une composition donnée. Si la fleur existe, la fonction renverra 1 sinon 0.

2. **Question 2**

   Ecrire 2 fonctions :

   Une permettant de connaître l'identifiant d'une composition en fonction de son code article, l'autre permettant de connaître l'identifiant d'une fleur en fonction de son code article.

3. **Question 3**

   En utilisant la requête écrite en 7, écrivez une procédure qui permettra de mettre à jour le prix d'une composition en fonction de celui des fleurs qui la composent.

   Cette procédure sera appelée avec un argument qui sera le code article de la composition.

4. Question 4

La procédure suivante permet d'enregistrer une nouvelle commande et de l'affecter au client concerné.

P.S. La fonction "println" est une fonction permettant de simplifier l'appel à DBMS_OUPUT.PUT_LINE().

```
CREATE OR REPLACE PROCEDURE "AQUIFLOR"."PASSERCOMMANDE"(idClient INTEGER, codeFleur VARCHAR := null, codeCompo VARCHAR := null,
    quantite INTEGER := 0, dteCde DATE := SYSDATE)
IS
    idFleur INTEGER := null;
    idCompo INTEGER := null;
    idCommande INTEGER := 0;
    clientExiste INTEGER := 0;
    mtCommande NUMBER(14,2) := 0;
BEGIN
    SELECT CASE WHEN COUNT(clt_id) > 0 THEN 1 ELSE 0 END INTO clientExiste
    FROM CLIENTS
    WHERE clt_id = idClient;
    IF (codeFleur IS NULL AND codeCompo IS NULL) THEN
        println('Vous devez sélectionner l''article à commander !');
    ELSIF (fonc_getIdFleur(codeFleur) = 0 AND codeFleur IS NOT NULL) THEN
        println('La fleur n''existe pas !');
    ELSIF (fonc_getIdCompo(codeCompo) = 0 AND codeCompo IS NOT NULL) THEN
        println('La composition n''existe pas !');
    ELSIF (clientExiste = 0) THEN
        println('Le client n''existe pas !');
    ELSIF (codeCompo IS NOT NULL AND codeFleur IS NOT NULL) THEN
        println('Une commande ne peut concerner qu''une composition ou une fleur !');
    ELSE

        IF (codeCompo IS NOT NULL) THEN
            idCompo := fonc_getIdCompo(codeCompo);
        ELSE
            idFleur := fonc_getIdFleur(codeFleur);
        END IF;
        INSERT INTO COMMANDES (cde_id, cde_date, cde_totalTTC, cde_idclient)
        VALUES(idCommande, dteCde, 0, idClient);

        INSERT INTO CONCERNER(conc_idcommande, conc_idfleur, conc_idcomposition, conc_quantite)
        VALUES (idCommande, idFleur, idCompo, quantite);
    END IF;
END;
```

Lors des tests, la première exécution n'a posé aucun soucis, par contre, la seconde a provoqué l'erreur :

```
>[Error] Script lines: 1-11 ------------------------
 ORA-00001: violation de contrainte unique (AQUIFLOR.SYS_C007012)
ORA-06512: à "AQUIFLOR.PASSERCOMMANDE", ligne 30
ORA-06512: à ligne 8
```

Selon vous, que signifie cette erreur et quelle en est la cause ?

Vous devez corriger cette procédure de manière à ce qu'elle fonctionne correctement.

Pour cela, vous avez la possibilité de choisir entre 3 solutions possibles :

- Ajouter une instruction SQL
- Créer un objet compteur et l'utiliser dans la procédure
- Créer un objet compteur et faire en sorte que son utilisation soit automatique de manière à ne pas modifier la procédure existante.

Vous devrez écrire une des 3 solutions.

5. Question 5

Compléter la procédure se trouvant à l'annexe D1 de manière à permettre la suppression d'une fleur dans une composition.
Une fleur ne peut-être retirée d'une composition que si elle y existe déjà.
Une composition ne contenant plus de fleurs ne doit plus exister !

6. Question 6

En vous servant de l'annexe E vous montrant comment travailler sur plusieurs enregistrements au sein d'une procédure (utilisation de curseurs), écrivez une procédure qui vous permettra d'obtenir le résultat suivant :

Année 2008 :

ADROUILLE-TOULTAN Yves
1 commande pour 1400 €
AJERRE Tex
1 commande pour 6785 €
AMARTAKALDIRE Quentin
1 commande pour 3120 €
ANCHIER Yvon
1 commande pour 10862,5 €
ARDELPIC Helmut
1 commande pour 58882 €
BLAIREUR Terry
4 commandes pour 30597 €

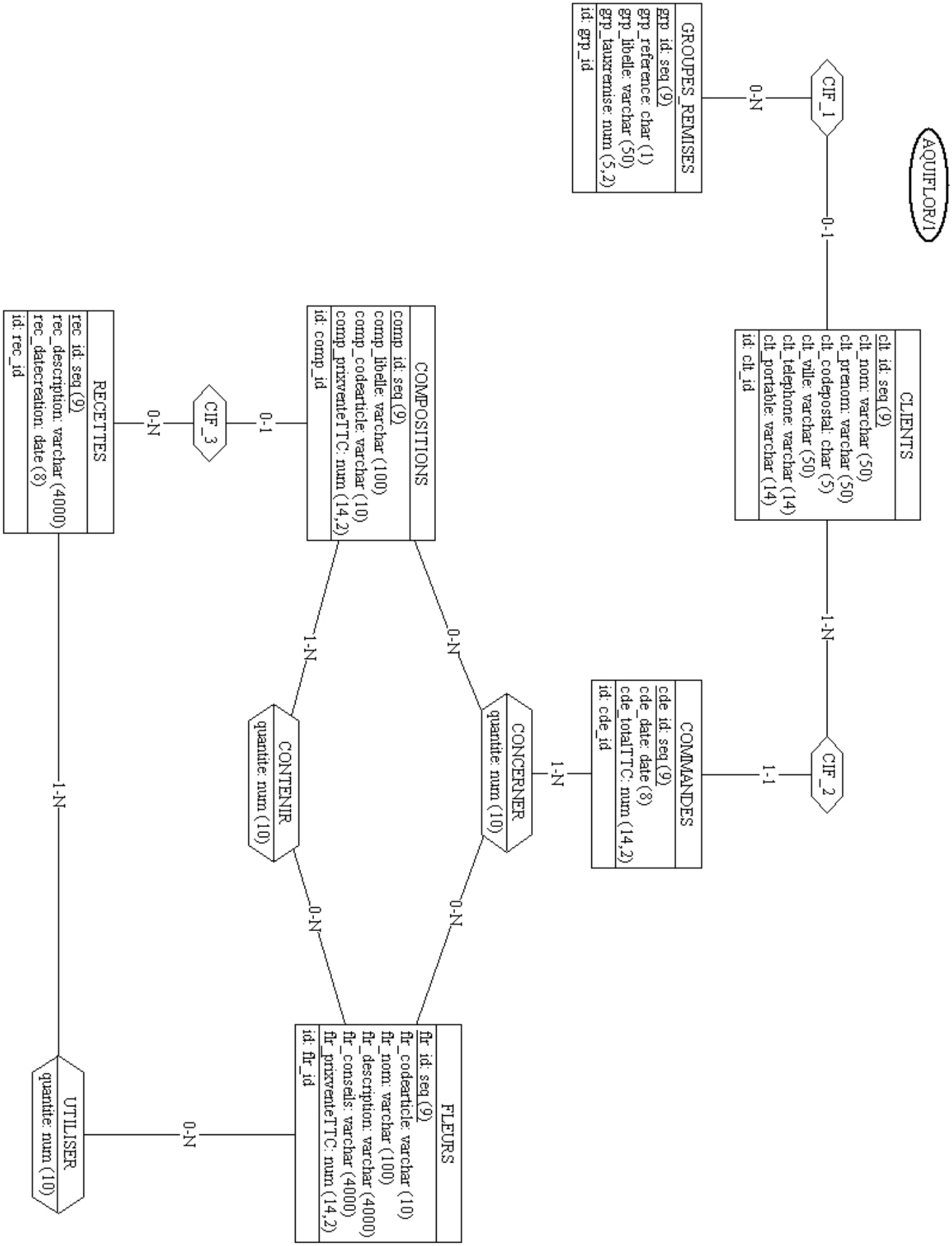
.
.
.

PATAMOB Adhémar
4 commandes pour 29771 €

Année 2007 :

.
.
.

Annexe A : MCD


AQUIFLOR/1
CLIENTS
clt_id: seq (9)
clt_nom: varchar (50)
clt_prenom: varchar (50)
clt_codepostal: char (5)
clt_ville: varchar (50)
clt_telephone: varchar (14)
clt_portable: varchar (14)
id: clt_id
GROUPES_REMISES
grp_id: seq (9)
grp_reference: char (1)
grp_libelle: varchar (50)
grp_tauxremise: num (5,2)
id: grp_id
CIF_1
0-N
0-1
CIF_2
1-N
1-1
COMMANDES
cde_id: seq (9)
cde_date: date (8)
cde_totalTTC: num (14,2)
id: cde_id
CONCERNER
quantite: num (10)
1-N
0-N
COMPOSITIONS
comp_id: seq (9)
comp_libelle: varchar (100)
comp_codearticle: varchar (10)
comp_prixventeTTC: num (14,2)
id: comp_id
CIF_3
0-N
0-1
RECETTES
rec_id: seq (9)
rec_description: varchar (4000)
rec_datecreation: date (8)
id: rec_id
CONTENIR
quantite: num (10)
1-N
0-N
FLEURS
flr_id: seq (9)
flr_codearticle: varchar (10)
flr_nom: varchar (100)
flr_description: varchar (4000)
flr_conseils: varchar (4000)
flr_prixventeTTC: num (14,2)
id: flr_id
UTILISER
quantite: num (10)
1-N
0-N

Annexe B1 : Exemples de données

Clients :

	CLT_ID	CLT_NOM	CLT_PRENOM	CLT_CODEPOSTAL	CLT_VILLE	CLT_TELEPHONE	CLT_PORTABLE	CLT_IDGROUPEREMISE
1	1	PATAMOB	Adhémar	33000	BORDEAUX	05-56-56-56-56	06-06-06-06-08	3
2	2	ZEUBLOUSE	Agathe	33000	BORDEAUX	02-41-33-31-18	06-08-06-08-70	1
3	3	KUZBIDON	Alex	33000	BORDEAUX	05-47-46-58-96	06-06-06-06-08	*(null)*
4	4	LOCALE	Anasthasie	33000	BORDEAUX	*(null)*	*(null)*	3
5	5	TEUTMARONNE	Armand	33000	BORDEAUX	*(null)*	*(null)*	3
6	6	ZOUDANLKOU	Debbie	33000	BORDEAUX	*(null)*	06-05-58-89-69	*(null)*
7	7	RIVENBUSSE	Elsa	33000	BORDEAUX	05-45-85-85-96	*(null)*	1
8	8	ARDELPIC	Helmut	33000	BORDEAUX	*(null)*	*(null)*	3
9	9	VRANTE	Héléna	33000	BORDEAUX	*(null)*	06-23-25-65-65	2
10	10	ENFAILLITE	Mélusine	33000	BORDEAUX	*(null)*	*(null)*	3
11	11	EURKETUMEME	Odile	33000	BORDEAUX	08-60-90-90-99	*(null)*	*(null)*
12	12	HOTDEUGOU	Olaf	33000	BORDEAUX	*(null)*	*(null)*	3
13	13	ODLAVIELLE	Pacôme	33000	BORDEAUX	*(null)*	06-58-56-25-05	1
14	14	AMARTAKALDIRE	Quentin	33000	BORDEAUX	*(null)*	*(null)*	2
15	15	PEURSCONLA	Humphrey	33000	BORDEAUX	*(null)*	*(null)*	3
16	16	TRAIBIEN	Samira	33000	BORDEAUX	*(null)*	*(null)*	3
17	17	FONFEC	Sophie	33000	BORDEAUX	*(null)*	*(null)*	3
18	18	FAIRANT	Teddy	33000	BORDEAUX	*(null)*	*(null)*	*(null)*
19	19	BLAIREUR	Terry	33000	BORDEAUX	*(null)*	*(null)*	3
20	20	AJERRE	Tex	33000	BORDEAUX	*(null)*	*(null)*	3
21	21	CHMONFISSE	Thierry	33000	BORDEAUX	*(null)*	*(null)*	3
22	22	PHOTOTETEDEMORT	Thomas	33000	BORDEAUX	*(null)*	*(null)*	3
23	23	KAECOUTE	Xavier	33000	BORDEAUX	*(null)*	*(null)*	*(null)*
24	24	ADROUILLE-TOULTAN	Yves	33000	BORDEAUX	*(null)*	*(null)*	1

Groupes de remises :

	GRP_ID	GRP_REFERENCE	GRP_LIBELLE	GRP_TAUXREMISE
1	1	A	Groupe A	5
2	2	B	Groupe B	12
3	3	C	Groupe C	24

Commandes :

	CDE_ID	CDE_DATE	CDE_TOTALTTC	CDE_IDCLIENT
1	1	17/09/2008 18:31:10	1348	2
2	2	17/09/2008 18:31:10	10862,5	5
3	3	17/09/2008 18:31:10	72331,84	4
4	4	17/09/2008 18:31:10	1348	13
5	5	17/09/2008 18:31:10	10862,5	15
6	6	17/09/2008 18:31:10	72331,84	22
7	7	17/09/2008 18:31:10	8968	1
8	8	17/09/2008 18:31:10	58882	8
9	9	17/09/2008 18:31:10	64008	12
10	10	17/09/2008 18:31:10	2296	15
11	11	17/09/2008 18:31:10	152633	16
12	12	17/09/2008 18:31:10	16933	1
13	13	17/09/2008 18:31:10	51240	17
14	14	17/09/2008 18:31:10	30	19
15	15	17/09/2008 18:31:10	75	19
16	16	17/09/2008 18:31:10	728	21
17	17	17/09/2008 18:31:10	490	21
18	18	17/09/2008 18:36:21	20	3
19	19	17/09/2008 18:36:21	8200	10
20	20	17/09/2008 18:36:21	5,7	18
21	21	17/09/2008 18:36:21	187,2	23
22	22	17/09/2008 18:36:21	10862,5	25
23	23	17/09/2008 18:36:21	1400	24

Annexe B2 : Exemples de données

Concerner :

	CONC_IDCOMMANDE	CONC_IDCOMPOSITION	CONC_IDFLEUR	CONC_QUANTITE
1	1	*(null)*	3	1685
2	2	*(null)*	6	9875
3	3	*(null)*	9	64582
4	4	*(null)*	3	1685
5	5	*(null)*	6	9875
6	6	*(null)*	9	64582
7	7	13	*(null)*	152
8	8	13	*(null)*	998
9	9	4	*(null)*	1524
10	10	1	*(null)*	164
11	11	13	*(null)*	2587
12	12	12	*(null)*	287
13	13	14	*(null)*	854
14	14	2	*(null)*	2
15	15	2	*(null)*	5
16	16	1	*(null)*	52
17	17	1	*(null)*	35
18	18	*(null)*	4	25
19	19	*(null)*	2	10000
20	20	*(null)*	7	5
21	21	*(null)*	5	240
22	22	*(null)*	10	9875
23	23	*(null)*	9	1250
24	24	9	*(null)*	32
25	25	10	*(null)*	52
26	26	11	*(null)*	95
27	27	12	*(null)*	115
28	28	13	*(null)*	185
29	29	2	*(null)*	258
30	30	16	*(null)*	365
31	31	5	*(null)*	456
32	32	2	*(null)*	756
33	33	15	*(null)*	725
34	34	4	*(null)*	689
35	35	*(null)*	4	1
36	36	*(null)*	2	1

Annexe B3 : Exemples de données

Compositions :

	COMP_ID	COMP_LIBELLE	COMP_CODEARTICLE	COMP_PRIXVENTETTC	COMP_IDRECETTE
1	10	60 Roses Equateur Variées	C_60RVE001	60	*(null)*
2	15	60 Roses Equateur Variées	C_60RVE006	59	*(null)*
3	14	60 Roses Equateur Variées	C_60RVE005	60	*(null)*
4	13	60 Roses Equateur Variées	C_60RVE004	59	*(null)*
5	12	60 Roses Equateur Variées	C_60RVE003	59	*(null)*
6	11	60 Roses Equateur Variées	C_60RVE002	59	*(null)*
7	16	Safran XXL - 51 Roses Equateur jaunes et oranges	C_SAFE0001	51	*(null)*
8	1	21 Roses Rouges	C_RR210001	14	*(null)*
9	2	21 Roses Blanches	C_RB210001	15	*(null)*
10	3	60 Roses Variées	C_60RV0001	43	*(null)*
11	4	60 Roses Variées	C_60RV0002	42	*(null)*
12	9	Safran XXL - 51 Roses jaunes et oranges	C_SAF00001	36	*(null)*
13	5	60 Roses Variées	C_60RV0003	42	*(null)*
14	8	60 Roses Variées	C_60RV0006	42	*(null)*
15	7	60 Roses Variées	C_60RV0005	42	*(null)*
16	6	60 Roses Variées	C_60RV0004	42	*(null)*

Fleurs :

	FLR_ID	FLR_CODEARTICLE	FLR_NOM	FLR_DESCRIPTION	FLR_CONSEILS	FLR_PRIXVENTETTC
1	1	RROUGE0001	Rose Rouge	Rose Rouge Locale	*(null)*	0,78
2	2	RBLANC0001	Rose Blanche	Rose Blanche Locale	*(null)*	0,82
3	3	RJAUNE0001	Rose Jaune	Rose Jaune Locale	*(null)*	0,8
4	4	RORANGE001	Rose Orange	Rose Orange Locale	*(null)*	0,8
5	5	RROSE00001	Rose Rose	Rose Rose Locale	*(null)*	0,78
6	6	REROUGE001	Rose Equateur Rouge	Rose à très gros boutons, classée comme la plus be…	*(null)*	1,1
7	10	REROSE0001	Rose Equateur Rose	Rose à très gros boutons, classée comme la plus be…	*(null)*	1,1
8	9	REORANGE01	Rose Equateur Orange	Rose à très gros boutons, classée comme la plus be…	*(null)*	1,12
9	8	REJAUNE001	Rose Equateur Jaune	Rose à très gros boutons, classée comme la plus be…	*(null)*	1,12
10	7	REBLANC001	Rose Equateur Blanche	Rose à très gros boutons, classée comme la plus be…	*(null)*	1,14

Contenir :

	CONT_IDFLEUR	CONT_IDCOMPOSITION	CONT_QUANTITE
1	1	1	21
2	2	2	21
3	3	3	30
4	4	3	30
5	1	4	30
6	5	4	30
7	1	5	20
8	5	5	20
9	3	5	20
10	1	6	20
11	5	6	20
12	4	6	20
13	5	7	20
14	3	7	20
15	5	7	20
16	1	8	15
17	5	8	15
18	3	8	15
19	4	8	15
20	3	9	25
21	4	9	26
22	8	10	30
23	9	10	30
24	6	11	30
25	10	11	30
26	10	12	20

Annexe C : Procédures Valides

AfficheComposition :

```
CREATE OR REPLACE PROCEDURE "AQUIFLOR"."AFFICHECOMPOSITION"(codeCompo VARCHAR)
AS
/*
    Fonction permettant d'afficher le détail d'une composition
    On part du principe que la composition demandée existe forcément et que des fleurs la composent
*/
    /* Déclaration d'une variable de type RECORD mappée sur la table COMPOSITION */
    enr_compo COMPOSITIONS%ROWTYPE;

    /* Déclaration du curseur de parcours des fleurs d'une composition */
    CURSOR c_fleurs (P_idCompo INTEGER) IS
            SELECT cont_quantite, flr_codearticle, flr_nom, flr_prixventeTTC
            FROM CONTENIR INNER JOIN FLEURS ON cont_idfleur = flr_id
            WHERE cont_idcomposition = P_idCompo;

    /* Déclaration des variables pour les fleurs */
    qteFlr INTEGER;
    codeFleur FLEURS.flr_codearticle%TYPE;
    nomFleur FLEURS.flr_nom%TYPE;
    prixFleur FLEURS.flr_prixventeTTC%TYPE;
BEGIN

    /* Affiche de l'enregistrement composition */
    SELECT * INTO enr_compo
    FROM COMPOSITIONS
    WHERE comp_id = fonc_getIdCompo(codeCompo);

    println(RPAD(enr_compo.comp_codearticle||' : '||enr_compo.comp_libelle||' ----- ', 50)||LPAD(enr_compo.comp_prixventeTTC, 15)||' €');

    /*Récupération des fleurs
        NOTE : Cette section utilise un curseur.
    */
    OPEN c_fleurs(fonc_getIdCompo(codeCompo));
    LOOP
        FETCH c_fleurs INTO qteFlr, codeFleur, nomFleur, prixFleur;
        EXIT WHEN c_fleurs%NOTFOUND;

        /* Affichage des informations */
        println(RPAD('     '||TO_CHAR(qteFlr)||' x '||codeFleur||' : '||nomFleur, 50)||LPAD(TO_CHAR(prixFleur), 15)||' € l''unité');
    END LOOP;
    CLOSE c_fleurs;
END;
```

AfficheToutesCompositions :

```
CREATE OR REPLACE PROCEDURE "AQUIFLOR"."AFFICHETOUTESCOMPOSITIONS"
AS
/*
    Fonction permettant d'afficher toutes les compositions
*/
    enr_comp COMPOSITIONS%ROWTYPE;
    CURSOR c_comp IS
        SELECT * FROM COMPOSITIONS ORDER BY comp_codearticle;
BEGIN
    /* Parcours du curseur */
    FOR enr_comp IN c_comp LOOP
        AfficheComposition(enr_comp.comp_codearticle);
        println('  ');
    END LOOP;
END;
```

Annexe D : Procédure à compléter (Question 7)

```
CREATE OR REPLACE PROCEDURE "AQUIFLOR"."EDITCOMPO"(act VARCHAR, codeCompo VARCHAR, codeFleur
VARCHAR, quantite INTEGER := 0)
AS
/* Procédure permettant de modifier le contenu d'une composition
   Param1 => Action :  ADD : Ajout d'une fleur,  DROP : Suppression d'une fleur, EDIT : Modification d'une fleur
   Param2 => codeArticle Composition
   Param2 => codeArticle Fleur
   Param3 => quantité
   La procédure affiche ensuite le contenu de la composition avec le prix de chacune des fleurs*/
   idCompo INTEGER;
   idFleur INTEGER;
   nbFleurs INTEGER;
   qteTemp INTEGER;
BEGIN
   IF (act IN ('ADD', 'DROP', 'EDIT')) THEN
      idCompo := fonc_getIdCompo(codeCompo);
      IF (idCompo = 0) THEN
         println('La composition n''existe pas !');
      ELSE
         idFleur := fonc_getIdFleur(codeFleur);
         CASE act
            WHEN 'ADD' THEN
               IF (fonc_fleurDansCompo(idCompo, idFleur) = 1) THEN
                  /* La fleur est déjà dans la composition, je rappelle la procédure pour ajouter la quantité */
                  SELECT cont_quantite INTO qteTemp
                  FROM CONTENIR
                  WHERE cont_idcomposition = idCompo
                     AND cont_idfleur = idFleur;

                  EditCompo('EDIT', codeCompo, codeFleur, (quantite + qteTemp));
                  RETURN;
               ELSE
                  INSERT INTO CONTENIR(cont_idcomposition, cont_idfleur, cont_quantite)
                  VALUES (idCompo, idFleur, quantite);
               END IF;
            WHEN 'DROP' THEN
               /*Suppression de la fleur désignée*/                    <---------- Compléter ICI
            WHEN 'EDIT' THEN
               IF (idFleur = 0) THEN
                  println('La fleur n''existe pas !');
                  RETURN;
               ELSE
                  IF (fonc_fleurDansCompo(idCompo, idFleur) = 0) THEN
                     /* Je rappelle la procédure pour ajouter la fleur */
                     EditCompo('ADD', codeCompo, codeFleur, quantite);
                     RETURN;
                  ELSE
                     UPDATE CONTENIR
                     SET cont_quantite = quantite
                     WHERE cont_idcomposition = idCompo
                        AND cont_idfleur = idFleur;
                  END IF;
               END IF;
         END CASE;
         AfficheComposition(codeCompo);
      END IF;
   ELSE
      println('L''action n''est pas définie !');
   END IF;
END;
```

Annexe E : Comment travailler sur plusieurs enregistrement dans une procédure PL/SQL

Une variable en PL/SQL ne peut recevoir qu'un seul enregistrement. De ce fait, pour travailler sur plusieurs enregistrements, il est nécessaire de savoir manipuler plusieurs enregistrements. Cela se fait par le biais d'un curseur.

Syntaxe :

Curseur Simple :

CURSOR *nomCurseur* IS *requete sql* ;

Curseur paramétré :

CURSOR *nomcurseur (parametres)* IS *requete sql* WHERE *condition* ;

Exemples :

Curseur simple :

```
SQL> Declare
  2    -- déclaration du curseur
  3    CURSOR C_EMP IS
  4    Select
  5       empno
  6      ,ename
  7      ,job
  8    From
  9      EMP
 10    ;
 11    -- variables d'accueil
 12    LN$Num EMP.empno%Type ;
 13    LC$Nom EMP.ename%Type ;
 14    LC$Job EMP.job%Type ;
 15  Begin
 16    Open C_EMP ; -- ouverture du curseur
 17    Loop -- boucle sur les lignes
 18      Fetch C_EMP Into LN$Num, LC$Nom, LC$Job ; -- Lecture d'une ligne
 19      Exit When C_EMP%NOTFOUND ; -- sortie lorsque le curseur ne ramène plus de ligne
 20    End loop ;
 21    Close C_EMP ; -- fermeture du curseur
 22  End ;
 23  /

Procédure PL/SQL terminée avec succès.
```

Curseur paramétré :

```
SQL> Declare
  2    -- déclaration du curseur
  3    CURSOR C_EMP ( PN$Num IN EMP.empno%Type )IS
  4    Select
  5       empno
  6      ,ename
  7      ,job
  8    From
  9      EMP
 10    Where
 11      empno = PN$Num
 12    ;
 13    -- variables d'accueil
 14    LN$Num EMP.empno%Type ;
 15    LC$Nom EMP.ename%Type ;
 16    LC$Job EMP.job%Type ;
 17  Begin
 18    Open C_EMP( 7369 ) ; -- ouverture du curseur avec passage du paramètre 7369
 19    Loop
 20      Fetch C_EMP Into LN$Num, LC$Nom, LC$Job ; -- Lecture d'une ligne
 21      Exit When C_EMP%NOTFOUND ; -- sortie lorsque le curseur ne ramène plus de ligne
 22      dbms_output.put_line( 'Employé ' || To_char(LN$Num) || ' ' || LC$Nom ) ;
 23    End loop ;
 24    Close C_EMP ; -- fermeture du curseur
 25    Open C_EMP( 7521 ) ; -- ouverture du curseur avec passage du paramètre 7521
 26    Loop
 27      Fetch C_EMP Into LN$Num, LC$Nom, LC$Job ; -- Lecture d'une ligne
 28      Exit When C_EMP%NOTFOUND ; -- sortie lorsque le curseur ne ramène plus de ligne
 29      dbms_output.put_line( 'Employé ' || To_char(LN$Num) || ' ' || LC$Nom ) ;
 30    End loop ;
 31    Close C_EMP ; -- fermeture du curseur
 32  End ;
 33  /
Employé 7369 SMITH
Employé 7521 WARD

Procédure PL/SQL terminée avec succès.
```