Corrigé de l'Examen de la 1ère session 2016-2017

Matière : Bases de données (SQL/PLSQL)		**Filière :** IIR **Niveau :** 3
Date de l'épreuve : 03 Février 2017	**Durée de l'épreuve :** 2H	
Documents autorisés Oui ☐ Non ☒x	Matériels autorisés : Oui ☐ : (Préciser)……. Non ☐	

Exercice

Considérons une base de données de gestion d'une partie des activités d'un fournisseur d'électricité (gestion des abonnements). La base de données (simplifiée) est décrite par les tables ci-dessous. Les clés primaires sont soulignées et les clés étrangères sont marquées par #.

Client (numClient, nomClient, prénomClient, adresse, ville, tél) ;

Compteur (numCompteur, marque) ;

Abonnement (numAbonnement, dateAbonnement, #numClient, #numCompteur, adresseAbonnement) ;

Facture (numFacture, dateFacture, #numAbonnement, ancienIndexCompteur, nouvelIndexCompteur, consommationEnkWh, dateDébutPériodeDeConsommation, dateFinPériodeDeConsommation, montant, dateLimiteDePaiement, Statut) ;

Paiement (numPaiement, #NumFacture, datePaiement, montantPayé) ;

TrancheDeConsommation (numTranche, borneInfConsommationTranche, borneSupConsommationTranche, tarifPourUnKWh) ;

- Un client peut souscrire plusieurs abonnements. Un compteur d'électricité est affecté à un et un seul abonnement.
Une facture correspond à un abonnement donné pour une période de consommation donnée.
- La consommation d'une facture est égale à la différence entre le nouvel index du compteur et l'ancien index.
- Le statut d'une facture non encore payée prend la valeur 0. Il passe à 1 une fois la facture payée.
- Pour chaque facture, le tarif d'un kWh d'électricité appliqué dépend de la tranche de consommation à laquelle appartient la consommation de la facture. Le tarif appliqué concerne la totalité de la consommation de la facture.

Exemple :
Trache 1 : 0 kWh -> 100 kWh ====> TarifkWh = 2 DH
Trache 2 : 101 kWh -> 200 kWh ====> TarifkWh = 3 DH
Trache 3 : 201 kWh -> 300 kWh ====> TarifkWh = 4 DH

Si Consommation de la facture = 75 kWh =====> Montant Facture = 75 x 2 DH = 150 DH
Si Consommation de la facture = 125 kWh =====> Montant Facture = 125 x 3 DH = 375 DH

Partie I : SQL (7 Points)

I.1) Ecrire les requêtes SQL qui permettent de créer les tables suivantes : *Client* et *Abonnement* (on suppose que les autres tables sont déjà crées). N'oubliez pas les contraintes d'intégrité clés primaires et clés étrangères **(1,5 Point)**

```
CREATE TABLE Client (numClient Number(4), nomClient Varchar2(30), prénomClient Varchar2(30),
adresse Varchar2(60), ville Varchar2(30), tél Varchar2(10), CONSTRAINT Pk_Client PRIMARY
KEY(numClient)) ;

CREATE TABLE Abonnement (numAbonnement Number(4), dateAbonnement Date, numClient
Number(4), numCompteur Number(7), adresseAbonnement Varchar(60), CONSTRAINT Pk_Abonnement
PRIMARY KEY(numAbonnement), CONSTRAINT Fk_Client FOREIGN KEY(numClient)
REFERENCES Client(numClient), CONSTRAINT Fk_Compteur FOREIGN KEY(numCompteur)
REFERENCES Compteur(numCompteur)) ;
```

I.2) Créer une vue permettant de consulter pour chaque abonnement le nombre de facture non payées en 2016. Quelles sont les opérations possibles via cette vue **(1,5 point)**

```
CREATE VIEW Factures_Non_Payees (numAbonnement, Nb_Factures_Non_Payees)
AS  SELECT numAbonnement, Count(*) FROM Facture  WHERE Statut = 0 and
TO_CHAR(dateFacture,'YYYY')='2016' GROUP BY numAbonnement;
```

La sélection (SELECT) est la seule opération permise sur cette vue (présence de COUNT et de Group By).

I.3) Lister les numéros de compteurs dont la marque commence par la lettre 'S' et se termine par la lettre 'M' **(1 Point)**

```
SELECT numCompteur FROM Compteur WHERE marque LIKE 'S%M' ;
```

I.4) Lister le nom du client et la marque du compteur associés aux abonnements effectués en 2016 **(1,5 Point)**

```
SELECT numAbonnement, nomClient, marque
FROM Client Cl, Abonnement A, Compteur Co
WHERE Cl.numClient=A.numClient AND A.numCompteur=Co.numCompteur
        AND TO_CHAR(dateAbonnement, 'YYYY')='2016' ;
```

I.5) Supprimer les abonnements ayant plus que 3 factures non payées en 2016 **(1,5 Point)**

```
DELETE FROM Abonnement A
WHERE (SELECT COUNT(*) FROM FACTURE F WHERE F.numAbonnement=A.numAbonnement
AND Statut=0 AND TO_CHAR(dateFacture,'YYYY')='2016') >3 ;
```

Partie II : PL/SQL (13 Points)

II.1) Ecrire une fonction *Recette_annuelle_Par_Client* qui prend comme paramètre le numéro d'un client et une année et retourne la recette annuelle réalisée par rapport à ce client (somme des

montants de toutes les factures du client en cette année). La fonction doit gérer l'erreur (exception interne) : Client inconnu.
(3 Points)

```
CREATE OR REPLACE FUNCTION Recette_annuelle_Par_Client(numCl IN Number, annee IN
Number) RETURN Number
V_Recette NUMBER ;
V_Nom Varchar2(30) ;

IS
BEGIN

SELECT nomClient INTO V_Nom
FROM CLIENT
WHERE numClient=numCl ;

SELECT SUM(montant) INTO V_Recette
FROM Facture F, Abonnement A
WHERE F.numAbonnement= A.numAbonnement
        AND A.NumClient =numCl
        AND TO_CHAR(dateFacture,'YYYY')=annee ;
RETURN V_Recette ;

EXCEPTION
WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Client inconnu') ;
END ;
/
```

II.2) Ecrire une procédure *Relevé_Abonnement_Par_Periode* qui prend trois paramètres : le numéro d'un abonnement, la date de début d'une période et la date de fin d'une période. Cette procédure affiche toutes les informations concernant les factures de cet abonnement pendant la période entre date de début et date de fin. La procédure doit aussi afficher le nombre de factures payées et le nombre de factures non payées. Elle doit également gérer deux erreurs (une exception externe et une exception interne) : date de début supérieur strictement à date de fin et numéro d'abonnement inconnu. **(4 Points)**

```
CREATE OR REPLACE  PROCEDURE Releve_Abonnement_Par_Periode(numAb IN Number,
dateDebut IN Date, dateFin IN Date)
IS
CURSOR C_Facture IS
SELECT * FROM Facture WHERE numAbonnement=numAb  AND dateFacture BETWEEN dateDebut
AND DateFin;

PERIODE_INVALIDE EXCEPTION ;

V_Ab Abonnement%ROWTYPE ;
CompteurFacturesPayees Number(7) :=0 ;
CompteurFacturesNonPayees Number(7) :=0 ;
```

```
BEGIN

IF dateDebut>DateFin THEN RAISE PERIODE_INVALIDE;  END IF ;


SELECT *  INTO V_Ab
FROM Abonnement
WHERE numAbonnement=numAb ;

FOR ENREG IN C_Facture LOOP
DBMS_OUTPUT.PUT_LINE('N° Facture : ' | | ENREG.numFacture| | '  Date Facture :' | |
ENREG.dateFacture| | '  Ancien index :' | | ENREG.ancienIndexCompteur| | '   Nouvel index :' | |
ENREG.nouvelIndexCompteur| | '   Consommation :' | |  ENREG.consommationEnkWh| | '   Date début
période : '| | ENREG.dateDébutPériodeDeConsommation| | '     Date Fin Période :' | |
ENREG.dateFinPériodeDeConsommation| | '    Montant :' | | ENREG.montant| | '    Date limite paiement :'
ENREG.dateLimiteDePaiement| | '    Statut : '| |  ENREG.Statut) ;

IF ENREG.Statut  = 1 THEN CompteurFacturesPayees= CompteurFacturesPayees+1 ;
ELSE CompteurFacturesNonPayees= CompteurFacturesNonPayees+1 ;
END IF ;
END LOOP ;

DBMS_OUTPUT.PUT_LINE('Nombre de factures payées  : ' | | CompteurFacturesPayees) ;
DBMS_OUTPUT.PUT_LINE('Nombre de factures non payées  : ' | | CompteurFacturesNonPayees) ;

EXCEPTION
WHEN NO_DATA_FOUND THEN DBMS_OUTPUT.PUT_LINE('Abonnement inconnu') ;
WHEN PERIODE_INVALIDE   THEN DBMS_OUTPUT.PUT_LINE('Période invalide : date début doit
être inférieure à date fin') ;

END ;
/
```

II.3) Ecrire un déclencheur qui change automatiquement le statut d'une facture non payée une fois le paiement de cette facture est effectué. (Insertion dans la table *Paiement*).
(2 Points)

```
CREATE OR REPLACE TRIGGER MAJ_Statut_Facture_trig
AFTER INSERT ON Paiement
FOR EACH ROW
BEGIN
UPDATE Facture
SET Statut=1
WHERE numFacture= :NEW.NumFacture ;
END ;
/
```

II.4) Ecrire un déclencheur qui calcule automatiquement le montant d'une facture en appliquant le tarif adéquat (selon la tranche de la consommation de la facture). **(2 Points)**

```
CREATE OR REPLACE TRIGGER Calcul_Montant_Facture_trig
BEFORE INSERT ON Facture
FOR EACH ROW
DECLARE
tarifApplique  number(5,2) ;
BEGIN
SELECT  tarifPourUnKWh    INTO tarifApplique
FROM TrancheDeConsommation
WHERE :NEW.consommationEnkWh  BETWEEN borneInfConsommationTranche AND
borneSupConsommationTranche ;

:NEW.montant :=  :NEW.consommationEnkWh  * tarifApplique ;

END ;
/
```

II.5) On suppose que tout paiement d'une facture effectué après la date limite de paiement de cette facture sera majoré par 20% du montant initial de la facture. Ecrire un déclencheur qui implémente cette règle de gestion (calcule automatiquement le montant à payer pour une facture en cas d'un paiement après la date limite). **(2 Points)**

```
CREATE OR REPLACE TRIGGER Calcul_Montant_Facture_Apres_DtLimite_trig
BEFORE INSERT ON Paiement
FOR EACH ROW
DECLARE
montantFactureAvantMajoration  number(11,2) ;
BEGIN
SELECT  montant INTO montantFactureAvantMajoration
FROM Facture
WHERE numFacture= :NEW.numFacture ;

:NEW.montantPaye:= montantFactureAvantMajoration*1,2 ;

END ;
/
```