EISTI Ecole Internationale des Sciences du Traitement de l'Information	**EXAMEN DE TRAITEMENT,** **ADMINISTRATION ET OPTIMISATION DES DONNEES** **Durée : 2 heures** **Tous les documents papiers sont autorisés**
Rédigé par : L'équipe pédagogique	Ref : *SE- TAD*
A l'intention de : Elèves ingénieurs 2ème année	Créé le : 10/11/2016

Nous considérons le schéma relationnel suivant modélisant les activités d'une agence de location de voitures. On suppose cette base créée dans le schéma « rapideloc » et que vous êtes administrateur.

*Vendeurs(**idvdr**, nom, prenom)*
*Clients (**idclt**, nom, adresse, nbvoitureslouees)*
*Voitures (**idvoit**, dateconstruction, daterevision, kilometrage,#idmarque)*
*Marques (**idmarque**, nomMarque)*
*Locations (**idloc**, dateloc, duree, KMdepart, KMretour, etat ,#idclt, #idvoit, #idvdr)*

L'état de location correspond à « en cours », « rendue » ou « retard ». Les kilomètres retour ne peuvent être remplis qu'au retour.

I.Administration Base de Données

1. Créer les rôles suivants : **1 pt par question**

a) **RVendeur** : il a tous les droits sur les tables clients et locations mais ne peut que consulter le reste

```
CREATE role RVendeur;
GRANT select, insert, update, delete ON rapideloc.Clients TO RVendeur;
GRANT select, insert, update, delete ON rapideloc.Locations TO RVendeur;
GRANT select ON rapideloc.Vendeurs TO RVendeur;
GRANT select ON rapideloc.Marques TO RVendeur;
GRANT select ON rapideloc.Voitures TO RVendeur;
```

b) **RClient** : il ne peut que consulter les tables voitures et locations

```
CREATE role RClient;
GRANT select ON rapideloc.Locations TO RClient;
GRANT select ON rapideloc.Voitures TO RClient;
```

c) **RComptable** : accès en lecture seule aux tables locations, voiture et marques.

```
CREATE role RComptable ;
GRANT select ON rapideloc.Locations TO RComptable;
GRANT select ON rapideloc.Voitures TO RComptable;
GRANT select ON rapideloc.Marques TO RComptable;
```

d) **RGaragiste** : accès total sur la table voitures. Il peut consulter toutes les autres tables.

```
CREATE role RGaragiste ;
GRANT select, insert, update, delete ON rapideloc.Voitures TO RGaragiste;
GRANT select ON rapideloc.Vendeurs TO RGaragiste;
GRANT select ON rapideloc.Clients TO RGaragiste;
GRANT select ON rapideloc.Marques TO RGaragiste;
GRANT select ON rapideloc.Locations TO RGaragiste;
```

2. Créer les vues suivantes et les droits qui vont avec :
   a) La liste des voitures (toutes informations) disponibles à la location, triée par marque. Cette vue devra être accessible aux clients et aux vendeurs. **2 pts**

```
-- comprend les voitures jamais louées et celles rendues
CREATE VIEW VoituresDispoLoc AS
SELECT nomMarque, v.idvoit, dateconstruction, daterevision, kilometrage
FROM Voitures v, Marques m
WHERE v.idmarque = m.idmarque
  -- requête imbriquée obligatoire car il peut y avoir des voitures jamais louées
  AND v.idvoit NOT IN (SELECT DISTINCT idvoit
                       FROM Locations
                       WHERE etat != 'rendu')
ORDER BY m.idmarque;

GRANT select ON rapideloc.VoituresDispoLoc TO RVendeur;
```

   b) La liste des clients (id et nom) ayant loué une voiture et dont la durée est dépassée. Cette vue ne sera accessible qu'aux vendeurs. **1 pt**

```
CREATE VIEW ClientsEnRetard AS
SELECT c.idclt, c.nom
FROM Clients c, Locations l
WHERE c.idclt = l.idclt
  AND sysdate > ADD_DAYS(dateloc, duree);
-- ou  AND etat = 'retard'

GRANT select ON rapideloc.ClientsEnRetard TO RVendeur;
```

   c) La liste des voitures à réviser, c'est-à-dire les voitures ayant parcouru plus de 50 000km et n'ayant pas été révisées dans les deux dernières années. Cette vue ne sera accessible qu'aux garagistes. **2 pts**

```
CREATE VIEW VoituresAReviser AS
SELECT v.idvoit, daterevision, SUM(KMretour - KMdepart) KMparcourus
FROM Voitures v, Locations l
WHERE l.idvoit = v.idvoit
  AND l.dateloc >= v.daterevision
  AND ADD_YEARS(daterevision, 2) < sysdate
GROUP BY v.idvoit, daterevision
HAVING KMparcourus > 50000 ;

GRANT select ON rapideloc.VoituresLoueesAnnee TO RGaragiste;
```

   d) La liste des voitures louées par les vendeurs dans l'année en cours, avec les durées de location. Cette vue ne sera accessible qu'au comptable pour l'établissement des primes des vendeurs. **1 pt**

```
CREATE VIEW VoituresLoueesAnnee AS
SELECT idvdr, nom, prenom, v.idvoit, duree
FROM Voitures vo, Locations l, Vendeurs ve
WHERE vo.idvoit = l.idvoit
  AND l. idvdr = ve.idvdr
  AND YEAR(dateloc) = YEAR(sysdate);

GRANT select ON rapideloc.VoituresLoueesAnnee TO RComptable;
```

II.PL/SQL

a) Les identifiants des 5 tables étant des entiers, créer les séquences pour les 5 tables. **1 pt**

```
CREATE sequence idvdr START WITH 1
CREATE sequence idclt START WITH 1
CREATE sequence idvoit START WITH 1
CREATE sequence idmarque START WITH 1
CREATE sequence idloc START WITH 1
```

b) Créer un trigger pour l'insertion d'une nouvelle location. Il devra valider les champs saisis et déclencher une erreur qui affichera un message si le champ n'est pas valide. En particulier, il devra capter l'inexistence du client, de la voiture ou du vendeur. La date devra être celle du jour. Le kilométrage sera automatiquement rempli comme égale au kilométrage de la voiture. La durée sera celle demandée et l'état sera positionné à « en cours ». Enfin, le champ nbvoitureslouees de la table Clients devra être mise à jour. **3 pts**

```
CREATE OR REPLACE TRIGGER newLocation
BEFORE INSERT ON Locations FOR EACH ROW
   err_sans_clt exception;
   err_sans_voit exception;
   err_sans_vdr exception;
BEGIN
   -- Validité des clé étrangères
   SELECT count(*) INTO cpt FROM Clients  WHERE idclt = :new.idclt;
   IF :new.idclt is null OR cpt = 0 THEN
         raise err_sans_clt;
   END IF;
   SELECT count(*) INTO cpt FROM Voitures WHERE idvoit = :new.idvoit;
   IF :new.idvoit is null OR cpt = 0 THEN
         raise err_sans_voit;
   END IF;
   SELECT count(*) INTO cpt FROM Vendeurs  WHERE idvdr = :new.idvdr;
   IF :new.idvdr is null OR cpt = 0 THEN
         raise err_sans_vdr;
   END IF;
   -- Remplissage des champs prédéfinis
   SELECT seqIdloc.nextval INTO :new.idloc FROM dual;
   IF :new.dateloc <> sysdate THEN
         :new.dateloc := sysdate;
   END IF;
   SELECT kilometrage INTO :new.KMdepart FROM Voitures WHERE :new.idvoit = Voitures.idvoit;
   :new.etat := 'en cours';
   UPDATE Clients SET nbvoiturelouees = nbvoiturelouees + 1 WHERE idclt = :new.idclt;
EXCEPTION -- Traitement des exceptions
   WHEN err_sans_clt THEN
         raise_application_error(-20001,'erreur parametre idclt');
   WHEN err_sans_voit THEN
         raise_application_error(-20002,'erreur parametre idvoit');
   WHEN err_sans_vdr THEN
         raise_application_error(-20003,'erreur parametre idvdr');
END;
/
```

c) Créer un trigger validant le retour d'une voiture louée. L'état passera à « rendu » et le kilométrage de retour sera validé. De plus, le kilométrage de la voiture devra être mise à jour. **2 pts**

```
CREATE OR REPLACE TRIGGER retourVoiture
BEFORE UPDATE ON Locations FOR EACH ROW
   err_sans_KMR exception;
BEGIN
   IF :new.KMretour is null OR :new.KMretour < :old.KMdepart THEN
          raise err_sans_KMR;
   END IF;
   :new.etat := 'rendue';
   :new.duree := sysdate - dateloc + 1;
   UPDATE Voitures SET kilometrage = KMretour WHERE Voitures.idvoit = :old.idvoit;
EXCEPTION
   WHEN err_sans_KMR THEN
          raise_application_error(-20004,'erreur parametre kilometrage');
END;
/
```

d) Définir une procédure ***dernièreLocation()*** qui affiche, pour chaque client, la date, le numéro et la marque de la dernière voiture louée. **2 pts**

```
CREATE OR REPLACE PROCEDURE dernièreLocation
IS
   CURSOR cur_derloc IS
          SELECT cl.idclt client, dateloc, idvoit, nomMarque
          FROM Clients cl, Locations l, Marques m
          WHERE cl.idclt = l.idclt
            AND l.idvoit = v.idvoit
            AND v.idmarque = m.idmarque
            AND dateloc >= (SELECT MAX(dateloc) FROM Locations WHERE Locations.idclt =
cl.idclt);
BEGIN
   dbms_output.put_line('Client :  client, date de location, voiture, Marque');
   FOR unClt IN cur_derloc
    LOOP
          dbms_output.put_line('Client : ' || unClt.client || ', ' || unClt.dateloc || ', '
|| unClt.idvoit || ', ' || unClt.nomMarque);
    END LOOP;
END dernièreLocation;
/
```

e) Définir une procédure ***verifLocation()*** qui met à jour la table location en passant l'état à « retard » lorsque la durée de location est dépassée. **1 pt**

```
CREATE OR REPLACE PROCEDURE verifLocation
IS
BEGIN
   UPDATE Locations SET etat = 'retard'
   WHERE sysdate > ADD_DAYS(dateloc, duree);
END verifLocation;
/
```