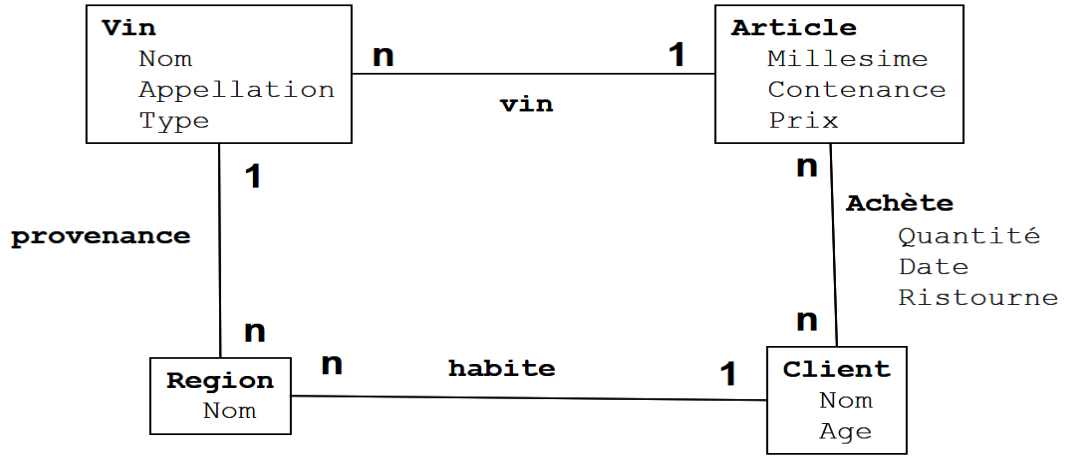
Triggers et Vues sous ORACLE
Travail en binôme

Le fichier **vin.sql** est à télécharger sur le campus. Il contient la définition des tables d'une base de données de vins, ainsi que des données sur l'application. Le modèle conceptuel de données (MCD) correspondant à cette base de données est proposé ci-dessous.



Veuillez charger cette base de données sous Oracle. Les tables créées dans **vin.sql** sont les suivantes :

```
Region (idregion, nom)
Vin (idvin, nom, appellation, type, provenance*)
Article (idarticle, millesime, contenance, prixht, vin*)
Client (idclient, nom, age, habite*);
Achat (idachat, article*, client*, dateachat, quantite, ristourne)
```

Tous les Triggers que vous proposerez doivent être testés par des requêtes si possible (au moins 2).

A. Création et gestion des Triggers sous ORACLE

1. Créer un trigger PL/SQL qui contrôle que toute insertion dans la table Client implique un âge supérieur à 18 ans.
2. Créer un trigger qui interdit d'acheter le Week End. Il interdit aussi la modification de la quantité ou bien l'insertion d'une quantité supérieure à 12.
3. Créer un trigger PL/SQL qui contrôle que toute insertion dans la table Article implique une contenance supérieure à 25 et toute mise à jour de la contenance doit être supérieure à la valeur dans la base.
4. Créer un trigger PL/SQL qui empêche d'effectuer une requête la nuit sur la table Vin. Entre 22h et 6h du matin !
5. Créer un trigger PL/SQL qui empêche d'annuler un achat ou de diminuer la quantité d'un article acheté ou d'augmenter la ristourne.
6. Créer un trigger qui interdit d'acheter le même article moins de 3 jours avant un achat effectué de cet article.

7. Soit la table d'audit ci-dessous qui permet de suivre les achats de vins.

```
CREATE TABLE AUDIT_ACHAT_VIN
    (NOM_CLIENT VARCHAR2 (30) NOT NULL,
    NOM_VIN VARCHAR2 (30) NOT NULL,
    NOM_REGION_VIN VARCHAR2(20) NOT NULL,
    APPELATION VARCHAR2 (30) NOT NULL,
    QUANTITE_COMMANDEE INTEGER,
    DATE_ACHAT DATE NOT NULL);
```

Écrire un trigger qui agit après une insertion dans la table ACHAT en remplissant la table **AUDIT_ACHAT_VIN**.

8. Proposer une méthode (Trigger ou autre technique) qui met à jour automatiquement la table suivante après chaque insertion dans la table Achat. **Track_Nombre_Client (idvin, vin_nom, vin_appelation, nombre_client).** L'attribut **nombre_client** doit pour chaque vin acheté, comptabiliser le nombre de clients différent ayant achetés ce vin.

B. Création et gestion des vues sous ORACLE

1. Nous souhaitons utiliser le mécanisme des vues pour créer la table « **AUDIT_ACHAT_VIN »** vue précédemment. Proposez ci-dessous la requête de création de vue pour cette table. Veuillez nommer la vue à créer « **AUDIT_ACHAT_VIN_View »**.
   a) Diminuer la quantité commandée de tous les Vins de 10 à partir de la vue créée.
   b) Cette modification sera-t-elle répercutée dans la table Achat ? Pourquoi ?
2. Créer une vue appelée **Regionale** qui affiche la jointure entre les régions et les clients. **Diminuer l'âge** de tous les clients de cette vue d'une année. Cette modification a-t-elle été répercutée dans la table Client ? Supprimer de la vue (mais pas de la base) toutes les lignes correspondant à un client de plus de 60 ans.
3. Créer une vue appelée **Stock** qui affiche la jointure entre les vins et les articles. À partir de cette vue, créer une autre vue appelée **Affaire** qui, pour chaque vin, donne le meilleur prix au litre possible. À partir de là, comment récupérer le vin qui correspond à la meilleure affaire ?
4. Créer une vue appelée **VinQuantité** qui pour chaque vin (idVin, nom, appellation), donne la quantité achetée à ce jour. En déduire le vin le plus acheté et le moins acheté. ?
5. Créer une vue qui pour chaque client affiche la quantité totale d'articles achetée à ce jour (tout article confondu). Créer une seconde vue qui affiche la quantité totale d'articles achetée par client et pour chaque article.
6. Créer une vue qui pour chaque article affiche la quantité totale achetée à ce jour (tout client confondu).
7. A partir des deux vues précédentes, en déduire les requêtes suivantes :

   - *nom, âge et région des clients qui achètent le plus de vin (en quantité)*
   - *nom, âge et région des clients qui dépensent le plus dans l'achat des vins*
   - *Quelle est la région qui achète le plus de vin en quantité*
   - *Quelle est la région qui vend le plus de vin en quantité*

C. Des fonctions, procédures et curseurs en PLSQL

1. Créer une fonction PL/SQL qui prend en paramètres l'identifiant d'un client et l'identifiant d'un article et qui renvoie le prix payé par le client pour l'achat de l'article. Tester cette fonction avec un client et un article existant dans la base de données.

2. Reprendre cette même fonction en considérant toutes les exceptions possibles : client inexistant, article inexistant, les exceptions prédéfinies *NO_DATA_FOUND et TOO_MANY_ROWS*. Tester.

3. En utilisant la fonction précédente, écrire une procédure qui affiche pour chaque achat :
   - *Le nom et l'appellation du vin acheté, le nom et la région du client et le prix payé.*

4. Dans un bloc PL/SQL anonyme, déclarer un curseur permettant de lire les données suivantes : *idarticle, idclient, dateachat, quantité*. Pour chaque achat lu par le curseur, afficher le nom du client, son âge et sa région, le nom et l'appellation du vin et la date de l'achat.

5. Il vous est demandé d'établir pour chaque client, le nombre d'achats (nbAchat) réalisés pour chaque vin existant. Vous devez gérer tous les cas possibles : un client sans achat, un client sans achat pour un vin existant, un vin jamais acheté. La procédure à écrire mettra à jour la table RESULTAT suivante :

   RESULTAT (idclient, nomClient, nomVin, nbAchat, prixTotal)

6. Ajouter la colonne ***saison*** dans la table achat. Cette colonne pourra avoir les valeurs suivantes : **printemps, été, automne, hiver**
   Créer une procédure qui met à jour tous les achats pour renseigner cette colonne.

7. Créer une procédure SQL qui augmente de n% (n en paramètre, compris entre 0 et 85 inclus) le prix de tous les articles de vins provenant d'une région **r** donnée en paramètre. Définissez les exceptions sur les valeurs de n et r.

8. Proposer une fonction puis une procédure qui réutilise cette fonction pour mettre en œuvre une fonctionnalité qui vous semble intéressante pour la gestion de cette base de données. Il s'agit ici de mettre en pratique votre capacité à proposer un service intéressant pour la gestion du SI.