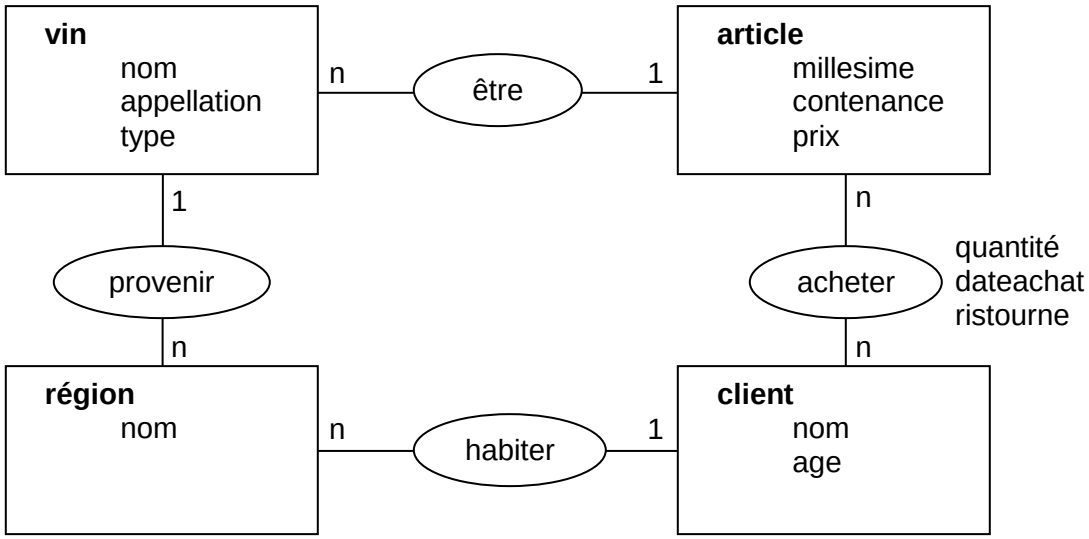
TP Oracle n°4

- Requêtes, vues, séquences -

1°) Base de données fournie

Le fichier `/users/prof/fernandes/data/vin.sql` contient la base de données correspondante au MCD ci-dessous.



Chargez le fichier `/users/prof/fernandes/data/vin.sql` sous Oracle. Les tables créées sont les suivantes. Remarquez, comme pour le TP précédent, la présence d'une table supplémentaire permettant de faire la gestion de la relation N-N entre les entités Article et Client.

Les tables suivantes sont donc créées :

région (idregion, nom)
vin (idvin, nom, appellation, type, **provenance**)
article (idarticle, millesime, contenance, prixht, **vin**)
client (idclient, nom, age, **habite**)
achat (idachat, **idarticle**, **idclient**, dateachat, quantite, ristourne)

En lançant la requête SQL `SELECT * FROM cat ;` vous pouvez vérifier que les tables ont bien été créées dans le catalogue.

La commande suivante permet de vérifier la présence de contraintes sur les tables :

```
SELECT constraint_name, constraint_type, table_name FROM user_constraints ;
```

2°) Requêtes

Quelles requêtes permettent de répondre aux questions suivantes ?
Sauvegarder l'ensemble des requêtes dans un fichier `tp4-1.sql`.

1. Lister tous les vins de la région 'Alsace', et, pour chacun des vins, le nombre d'articles correspondant.
2. Indiquer le prix maximum, minimum, moyen des vins en général, puis par région.
3. Lister, pour chaque client, le vin le moins cher en 75cl disponible dans la région.
4. Lister les vins disponibles en 150cl, ainsi que leur prix et leur millésime.
5. Lister les appellations dont certains des vins disponibles sont antérieurs à 1992. Lister, avec chacune de ces appellations, le prix maximum des vins et l'année du vin en question.
6. Pour chaque appellation, lister le plus vieux des vins rouges.

3°) Création de vues

Définition : Une vue permet d'assurer l'indépendance logique. Grâce à elle chaque utilisateur pourra avoir sa propre vision des données. L'utilisateur pourra alors consulter (et modifier avec certaines restrictions) la base à travers la vue, c'est-à-dire avec la commande SELECT comme s'il s'agissait d'une table.

Syntaxe simplifié de création / remplacement:

```
CREATE [OR REPLACE] VIEW <nomvue> (<alias>,...) AS <Requête> [WITH READ ONLY] ;
```

Syntaxe pour la suppression d'une vue:

```
DROP VIEW <nomvue> ;
```

1. Créer une vue appelée Regionale qui affiche la jointure entre les régions et les clients. Augmenter 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 40 ans.
2. Créer une vue appelée Stock qui affiche la jointure entre les vins et les articles. A 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 les articles qui sont cette meilleure affaire ?

4°) Création de séquences

Définition : Une séquence génère des entiers uniques qui appartiennent à une suite arithmétique. Elle permet en général de remplir la clé primaire automatiquement.

Syntaxe de création d'une séquence:

```
CREATE SEQUENCE <nomseq> [INCREMENT BY i] [START WITH deb] ;
```

Par défaut `i` vaut 1 et `deb` vaut 0.
Pour accéder à la valeur suivante de la séquence, on utilise `nomseq.nextval`. Avant la première utilisation d'une séquence, il faut l'initialiser avec la commande :

```
SELECT <nomseq>.nextval FROM dual ;
```

La valeur actuelle de la séquence peut être récupérée avec la commande :

```
SELECT <nomseq>.currval FROM DUAL ;
```

Créer une séquence pour les clients et insérer un nouveau client.