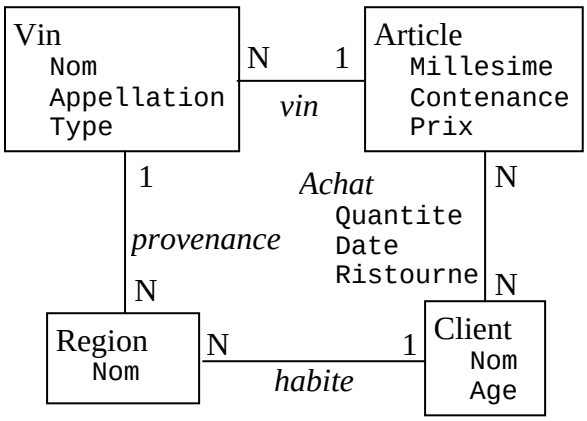
Université de Provence
Licence d'Informatique
Troisième année

Bases de données
TP No 6
SQL_PLUS d'Oracle
PL/SQL – Curseurs - Triggers

1) Base de données fournie

Le fichier `vin.sql`, que vous trouverez sur www.cmi.univ-mrs.fr/~contensi/BD05/vin.sql, contient la définition des tables d'une base de données de vins, ainsi que quelques données. Le modèle conceptuel de données (MCD) correspondant à cette base de données est donné ci-contre.



Charger cette base de données sous Oracle. Les tables créées dans `cinema.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, #idarticle, #idclient, date, quantite, ristourne)
```

2) Compréhension de blocs PL/SQL

Comprendre puis lancer l'exécution du bloc PL/SQL ci-dessous (utilisez la documentation *Oracle PL/SQL* chap. 1 & 7 et éventuellement la documentation *Oracle SQL* chap. 4 pages 232-236).

```
CREATE FUNCTION num_auto_vin RETURN NUMBER IS
Numax NUMBER;
BEGIN
   SELECT MAX (idvin) INTO numax FROM Vin;
   RETURN numax+1;
END;
```

3) Premières créations de blocs avec gestion des exceptions

1. Créer une fonction PL/SQL qui prend en paramètre un nom de vin et renvoie l'identifiant de ce vin s'il existe, et 0 sinon (cf. chap. 6 sur les exceptions, utilisez `OTHERS`). Pour tester la fonction, utilisez :
   `SELECT nom_fonct(argument) FROM DUAL;`
2. Supprimez cette fonction et recréez-la en utilisant les exceptions prédéfinies `NO_DATA_FOUND` et `TOO_MANY_ROWS`.

4) Curseurs

Les curseurs sont des pointeurs sur une zone mémoire pour les données extraites de la base. Il existe des curseurs implicites et explicites. Oracle ouvre toujours un curseur implicite pour traiter une instruction SQL, celui-ci ne se rapporte qu'à la dernière instruction SQL exécutée et il se nomme « SQL ». Le curseur contient des attributs (`%NOTFOUND`, `%FOUND`, `%ROWCOUNT`) qui fournissent des informations sur l'exécution des instructions `INSERT`, `UPDATE`, `DELETE`, `SELECT INTO`. Un curseur implicite pour une instruction `SELECT INTO` ne peut gérer qu'une seule ligne. Le curseur explicite quant à lui place le résultat d'une requête multi-lignes dans un tampon mémoire et libère les lignes les unes après les autres lors du traitement.

Le curseur se définit dans la partie déclarative du bloc PL/SQL (la requête n'est pas exécutée à ce moment-là). Dans cette déclaration, il est possible de donner une clause `FOR UPDATE OF nom_colonne(s)` qui permet de verrouiller les lignes sélectionnées (aucun autre utilisateur ne peut mettre à jour tant que le verrou n'est pas retiré). La commande `OPEN nom_curseur;` exécute la requête et place le curseur en mémoire, elle ne retourne aucun résultat. L'instruction `FETCH nom_curseur INTO variable;` extrait la ligne courante du curseur, la place dans une variable et fait avancer le curseur à la ligne suivante. Pour parcourir toutes les lignes du curseur, il faut utiliser une boucle `LOOP`. La clause `CURRENT OF nom_curseur` est utilisée dans le `WHERE` d'une commande `UPDATE` pour modifier la ligne courante (si un `FOR UPDATE` a été utilisé préalablement). L'arrêt de la boucle est obtenu grâce à `nom_curseur%NOTFOUND` qui retourne false s'il ne reste plus de lignes. Pour libérer l'espace mémoire, il faut fermer explicitement le curseur en utilisant `CLOSE nom_curseur;`.

Un curseur peut accepter des paramètres en entrée, ils servent à passer des informations au curseur et sont généralement utilisés dans un `WHERE` pour limiter la requête. Les paramètres ont un type associé qui ne peut pas avoir d'indication de longueur. Ils sont passés lors de la commande `OPEN`.

1. Créer une procédure SQL qui augmente de `n%` (`n` en paramètre, compris entre 0 et 100 inclus) le prix de tous les articles antérieurs à une date `d` en paramètre. Définissez une exception dans le cas où le paramètre n'est pas compris entre 0 et 100.
2. Ajouter la colonne `stock` dans la table `Article`. Créer une procédure qui met à jour tous les articles pour renseigner cette colonne. Les articles antérieurs à 1950 sont uniques en stock. Les articles entre 1950 et 1970 sont au nombre de 4. Les articles entre 1970 et 1998 sont au nombre de 10. Tous les articles postérieurs à 1998 sont au nombre de 25.

5) Triggers

Les triggers permettent de gérer des événements de la base de données. Ils sont définis dans la base de données pour une table spécifique et se déclenchent lorsqu'il y a modification de données pour cette table (`INSERT`, `DELETE`, `UPDATE`). Les triggers contiennent du code PL/SQL, ils peuvent appeler toute procédure ou fonction stockée et peuvent référencer d'autres objets de la base de données. Un trigger peut être défini au niveau instruction (il s'active une fois pour une instruction SQL), ou au niveau ligne (il s'active une fois pour chaque ligne affectée par l'instruction et permet de référencer les données d'une ligne). Le trigger peut être activé avant que la ligne ne soit affectée (`BEFORE`) ou après (`AFTER`). Attention, les triggers ne doivent pas remplacer les contraintes d'intégrité !

Un trigger est stocké dans la base de données avec la commande :

```
CREATE TRIGGER nom_trigger BEFORE|AFTER
INSERT|DELETE|UPDATE [OF nom_col1,…,nom_colN] ON nom_tab [FOR EACH ROW];
```

La liste des colonnes ne sert que pour un trigger `UPDATE`. La clause `FOR EACH ROW` indique un trigger de niveau ligne. Le code source est mémorisé dans la table `USER_TRIGGERS`. Pour référencer les valeurs de colonnes dans des triggers de niveau ligne, il faut utiliser les identificateurs `:NEW` et `:OLD`. Le premier n'est disponible que pour `INSERT` ou `UPDATE`, le second n'est disponible que pour `UPDATE` ou `DELETE`.

1. Créer un trigger PL/SQL qui empêche d'effectuer une requête la nuit sur la table `Vin`.
2. Créer un trigger PL/SQL qui ajoute automatiquement une nouvelle valeur de clé primaire lorsqu'un nouveau vin est entré.