Chapitre 1 : Oracle - Premières notions

1.1 - Présentation

(Voir "Oracle 7" de A. Abdellatif, M. Limane, A. Zeroual p.9-11)

L'historique d'Oracle commence avec la création d'Oracle corporation en 1977.
La première version est commercialisée en 1979. C'était l'un des premiers SGBD Relationnels.
Une version PC voit le jour en 1984. Les versions Unix se succèdent.
Actuellement Oracle est un véritable environnement de travail avec des outils pouvant être classés
en outils de développement d'applications (seul point qui nous intéresse ici),
en outils de gestion de données réparties (accès à des données situées sur un site distant, partage de données entre plusieurs bases de données)
et en outils d'administration des données.

- **Oracle est un SGBDr** (Système de Gestion d'une Base de Données Relationnel)
  (DBMS : DataBase Management System)
  Architecture ANSI/SPARC : groupe de normalisation qui en 1975 a proposé 3 niveaux de description des données (interne, conceptuel, externe).
  (voir "Oracle 7" pages 9-11)

- **Les Objectifs d'un SGBDr**
  - Définition des données,
  - Manipulation des données,
  - Intégrité et Sécurité des données,
  - Transactions, accès concurrents
  - Reprises après panne.

- **Les notions importantes** :
  - les **schémas de relations**
    Les relations sont des *tables*.
    La table est un *objet* de la Base de Données (BD). Elle a
    - un nom,
    - une structure définie par le nom des colonnes (les propriétés de la relation) et, pour chaque colonne, le domaine des valeurs
    - un contenu : les lignes correspondent aux occurrences (ou n-uplets ou tuples) de la relation.
    Les structures et les données sont manipulables par des *opérations*.
  - les **opérations** de définition ou manipulation des données
    Ce sont des actions permettant aux utilisateurs de manipuler les données et les structures tout en respectant un ensemble de *contraintes d'intégrité*.
  - les **contraintes d'intégrité**
    Ce sont des règles, dites règles de gestion.
    Elles protègent les données et les structures.

- **Architecture fonctionnelle d'Oracle**
  (voir schéma dans "Oracle 7" p.33)
  - **les couches de base**
    - Le noyau : stockage physique des données, différents accélérateurs ou optimiseurs d'accès aux données,
      fonctions de vérification des contraintes d'intégrité et de cohérence des données, contrôle des accès concurrents,
      gestion de la confidentialité des données, reprise après panne.
    - Le dictionnaire des données est une "méta-base" qui décrit dynamiquement la base de données :

      . les objets (tables, colonnes, vues, index, ...)
      . les utilisateurs (nom, mots de passe)
      . les privilèges et les droits des utilisateurs sur les objets.

- La couche SQL : langage de commandes

  . langage de définition des données
  . langage de manipulation des données

- La couche PL/SQL : extension procédurale du langage SQL qui ne l'est pas.
  Avec SQLplus on peut :
  - Entrer, éditer, mémoriser, exécuter des commandes SQL et des blocs de commandes en PL/SQL
  - Mettre en forme des résultats de requêtes
  - Afficher la structure d'une table (définitions des colonnes, contraintes)
  - Saisir des valeurs au clavier
  - Faire des copies de tables entre différentes bases.

- **les outils**

  Nous n'en citerons que quelques-uns :
  - Les outils de développement d'applications : SQL*Plus, PRO*
  - Les utilitaires : SQL*Loader, SQL*dba
  - Les outils de communication
  - Les outils d'aide à la décision

1.2 - Connexion et accès à SQL*Plus

- **Connexion à SQL*plus :**
  prompt>**. oraenv** (Entrée)
  prompt>**sqlplus** (Entrée)
  login (entrer votre login) (Entrée)
  password (entrer votre mot de passe) (Entrée)
  SQL>

- **Déconnexion de SQL*plus :**
  SQL>**exit** (Entrée)
  prompt>

1.3 - Manipulations de base

- Aide en ligne

  help <*nom-commande*> : aide sur la commande citée (ex. : help exit)

- Dialogue avec le système hôte

  host <*commande*> : lance la commande qui doit être une commande du système d'exploitation.
  ex. : host ls (Entrée)

- Travail sous sqlplus
  1) **Travail en direct** :

  a) La plupart des commandes sql peuvent être tapées sur une ou plusieurs lignes et doivent être terminées par **;**
  SQL ne fait pas la différence entre minuscules et majuscules.
  Il est toutefois recommandé de mettre les mots SQL en majuscules, d'aller à la ligne et d'indenter les lignes de commande.

  b) Oracle interprète la commande et l'exécute immédiatement. Les résultats sont affichés.

  2) **Travail à partir d'un script**

  a) On écrit une suite de commandes (terminées par un ;) à l'aide d'un éditeur de texte (vi, emacs, xemacs, ...), bien qu'il y ait un buffer SQL et l'on sauvegarde le fichier dans le répertoire de travail, le nom du fichier devant avoir pour extension **.sql**

  b) On exécute les commandes soit en faisant du "couper/coller" de l'éditeur vers SQL, soit en dialogant avec le système hôte de la manière suivante :

start <*nomprog*> : charge le fichier et lance les commandes contenues dans le fichier *nomprog.sql* (inutile de taper .sql)

@ <*nomprog*> : charge le fichier et lance les commandes contenues dans le fichier *nomprog.sql* (ne pas taper .sql)

- **Exercice**
  - Sous un éditeur taper :
  - PROMPT Bonjour a tous!
  - -- ceci est une remarque se terminant à la fin de la ligne
  - REM c'est encore une remarque se terminant à la fin de la ligne
  - /*
  - ceci est un bloc de remarques
  - pouvant s'étendre sur plusieurs lignes
    */
  - Sauvegerder, puis exécuter.
  - Attention! Pour simplifier le travail, sauvegarder dans le répertoire dans lequel on a lancé Oracle (sqlplus).

1.4 - Tables : création, modification, suppression

Les principales notions concernant ces opérations sont décrites dans un aide-mémoire auquel on peut accéder en cliquant ici .

Remarque :
Les opérations Ajout, Modification, Suppression ne sont pas effectives automatiquement.
La commande **COMMIT** rend effectives les opérations faites précédemment (automatique lorsqu'on quitte Oracle).
La commande **ROLLBACK** permet de restaurer les tables dans l'état antérieur.

1.5 - Types de données

CHAR(n)	Chaîne de n caractères (0 < n < 256) Longueur par défaut 1 Longueur fixe : ajout de blancs ou troncature.
VARCHAR2(n)	Chaîne de caractères de longueur variable n. (0 < n < 2001)
VARCHAR(n)	Synonyme de VARCHAR2(n)
LONG	Chaîne d'au plus 2^{31}-1 caractères (~2 Go) Cela permet de mémoriser de longs textes. Mais : - une seule colonne de type LONG par table, - problème dans les requêtes
NUMBER(p,s)	Entier relatif ou réel la valeur absolue entre $1.0*10^{-130}$ et $9,9...9*10^{125}$ au plus 38 chiffres de précision p : nombre total de chiffres => précision s : nombre de décimales (-85 < s < 128) => échelle (*scale*) . si s est absent, c'est un **entier** . s peut être négatif, on a alors un arrondi ex. : si NUMBER(7,-3), 4253,56 devient 4000
DATE	Date entre le 1er janvier -4712 et le 1er décembre +4712 Sont stockés : siècle, année, mois, jour, heure, minute, seconde Une date est saisie sous forme de chaîne de caractères (conversion) En standard : 'DD-MON-YY' (en Anglais : 2 chiffres pour le jour, 3 pour le mois et 2 pour l'année)
LONGRAW	Donnée de type binaire de longueur variable pouvant aller jusqu'à 2 Go
ROWID	Chaîne hexadécimale représentant l'adresse unique d'une ligne de la table.

	Cette valeur est stockée dans la pseudo-colonne de même nom.

1.6 - Tables : Contraintes d'intégrité

Une **contrainte d'intégrité** est une règle qui permet de contrôler la validité et la cohérence des valeurs entrées dans les tables.

- **Contrainte de colonne** :
  Cette contrainte est ainsi nommée parce qu'elle concerne une colonne, elle est déclarée sur la ligne de définition de la colonne à laquelle elle se rapporte.
  Pas de virgule entre la définition de la colonne et celle de la contrainte

  [CONSTRAINT *nom_contrainte*] la_contrainte [option]

  Exemple :

  AGE NUMBER(3,0) CONSTRAINT Toujours NOT NULL,

  [CONSTRAINT *nom_contrainte*] : permet de donner un nom à la contrainte ; ce nom est sauvegardé dans le dictionnaire des données.

  la_contrainte :
  NOT NULL : spécifie que pour toute occurrence la colonne doit être valorisée

  UNIQUE : toutes les valeurs de la colonne sont nécessairement distinctes

  PRIMARY KEY : la colonne est une clé primaire pour la table.

  peut être remplacé par UNIQUE NOT NULL

  REFERENCES *table*(*colonne*) : il s'agit d'une contrainte d'intégrité fonctionnelle par rapport à une clé ; chaque valeur de la colonne doit exister dans la table et la colonne référencées.
  On utilise cette contrainte pour les clés étrangères.

  Dans une table COMMANDE,

  code_client VARCHAR2(8)
  CONSTRAINT fk_client REFERENCES CLIENT(code_client),

  A chaque entrée d'une valeur code_client dans la table COMMANDE, Oracle vérifie l'existence de cette valeur dans la table CLIENT.

CHECK : mot-clé associé à une condition qui doit être vérifiée par toutes les valeurs de la colonne (domaine des valeurs de l'attribut).

age NUMBER(3,0) CONSTRAINT Toujours NOT NULL CHECK (age BETWEEN 0 AND 150),

[option]

ON DELETE CASCADE : En sa présence, lorsqu'une occurrence est supprimée (suppression d'une instanciation de la clé primaire), dans chaque table ayant une clé étrangère associée, il y a suppression des occurrences correspondantes (même valeur de clé étrangère que la valeur de la clé primaire)

Dans l'exemple précédent, la suppression d'un client (à partir de son code)

entraînera la suppression de toutes les commandes de ce client

Remarque : L'utilisation des contraintes d'intégrité fonctionnelle impose un ordre dans la création des tables et dans leur suppression.

- **Contrainte de table** :
Une contrainte de table est une contrainte d'intégrité qui porte sur un ensemble de colonnes d'une même table.
Une virgule sépare la définition d'une contrainte de table des définitions des colonnes

[CONSTRAINT *nom_contrainte*] la_contrainte [option] :

[CONSTRAINT *nom_contrainte*] : permet de donner un nom à la contrainte ; ce nom est sauvegardé dans le dictionnaire des données.

UNIQUE (col i, col j, ...) : l'unicité porte sur le n-uplet des valeurs

PRIMARY KEY (col i, col j, ...) : clé primaire de la table

```
CREATE TABLE article_commande
 (no_commande NUMBER(5), -- numéro de la commande
  code_article VARCHAR2(10), -- référence de l'article commandé
  quantite NOMBER(3), -- nombre d'articles pour la référence et la commande
  CONSTRAINT C_artcom PRIMARY KEY (no_commande,code_article)
 );
```

FOREIGN KEY (col i, col j, ...) REFERENCES table.(col m, col n, ...)

ON DELETE CASCADE

désigne une clé étrangère constituée de plusieurs colonnes.

Il est très utile de mettre des commentaires pour la définition de chaque colonne, mais il ne faut pas en mettre sur la dernière ligne contenant);

- # 1.7 - Consultation de données

Oracle - Consultation des données

- Voir le contenu d'une table (titre des colonnes et occurrences)

  SELECT * FROM T ;
  => affichage du contenu de la table T

- Restreindre la consultation à certaines colonnes de la tables

  SELECT c1, c2, ..., ck
  FROM T;
  => seules les colonnes c1, c2, ..., ck sont affichées
  dans l'ordre dans lequel elles ont été demandées.

  On peut intercaler ALL ou DISTINCT entre la commande SELECT et le nom des colonnes.
  SELECT [ALL/DISTINCT] c1, c2, ..., ck
  FROM T;
  ALL [par défaut] permet d'afficher toutes les occurrences (lignes).
  DISTINCT n'affiche que les occurrences (n-uplets, lignes) distinctes.

- **Remarque** : SELECT est une commande très importante.

  - C'est elle qui permet d'accéder aux données
  - Elle permet d'obtenir bien plus que des données brutes.
  - Le résultat de la sélection est une table

Quelques éléments du dictionnaire des données

1) Dans le Dictionnaire des Données, la table **all_catalog** contient le catalogue général de la BD.

• Structure de la table (nom et type des colonnes) :

DESC all_catalog ;

• Contenu de la table (toutes les occurrences) :

SELECT *

FROM all_catalog;

2) Liste des noms de tables de l'utilisateur (colonne table_name de la table user_tables) :

SELECT table name
FROM user tables;

3) Liste des noms de tables auxquelles l'utilisateur peut accéder avec le nom de leur propriétaire (colonnes owner, table_name de la table all_tables) :

SELECT owner, table_name

FROM all_tables ;

Remarque : Accès d'un utilisateur au dictionnaire des données

-- "USER_" : vues permettant de ne voir que les objets appartenant à un utilisateur,
-- "ALL_" : vue permettant de voir tous les objets accessibles à un utilisateur.

-- Exemple :

```
select table_name FROM user_tables;
select table_name FROM tabs; -- tabs est un synonyme de user_tables
select table_name FROM all_tables;
```

4) Création de tables à partir des tables fm.A_xxx :

Ici "fm" est le nom du propriétaire de la table et "A_xxx" est le nom de la table.

Exemple de copie d'une table avec sa structure et son contenu :

```
CREATE TABLE A_xxx
AS SELECT *
FROM fm.A_xxx ;
```

5) Travail sur une table :

-- afficher la structure d'une table :

Table dont fm est est le propriétaire et à laquelle l'utilisateur a accès :

```
DESC  fm.A_artiste
```

Table personnelle :

```
DESC A_artiste
```

Remarque : l'écriture de cette commande tient toujours sur une seule ligne,
elle n'a pas besoin du ; de fin de commande.

-- lister les nom, localite, datns (date de naissance) de tous les artistes :

-- projection sur plusieurs colonnes

```
SELECT nom, localite, datns
FROM A_artiste ;
```

6) Autorisations :

- donner une autorisation de lecture d'une table à un autre utilisateur

```
GRANT select       -- insert, update, all
ON nomtable
TO nom_utilisateur     -- ou TO PUBLIC
[WITH GRANT OPTION] ;
```

/*

la dernière ligne optionnelle permet à nom_utilisateur de transmettre à quelqu'un d'autre

l'autorisation qu'il a reçue.

*/

- retirer une autorisation :

```
REVOKE select
ON nomtable
FROM nom_utilisateur ;
```

/*
il y a suppression des droits en chaîne à ceux à qui l'utilisateur avait pu transmettre cette autorisation
*/

- # create table 1.8 - Exercices

TD/TP SQL, SQL*PLUS N°01

TD/TP SQL, SQL*PLUS N°01

CREATION D'UNE TABLE, MODIFICATION DE LA STRUCTURE, INSERTION DE DONNEES
---

1. Créer une table permettant de stocker des noms, prénoms, dates de naissance
(type date), taille en cm.

2. Structure de la table
- voir la structure à l'aide de la commande
  DESC[RIBE] nom_table
- modifier cette structure selon les règles suivantes :
  . la date de naissance est une chaine de 10 caractères 'DD/MMM/YYYY'
  . on ajoute le caractère poids (l'unité est le kg).

3. Contenu de la table
- insérer des occurrences
- visualiser le contenu de la table
- renommer la table

4. Voir la liste des tables personnelles
  SELECT table_name from tabs;

5. Suppression de la table

GESTION DE VENTE À DOMICILE

On se place dans le cadre d'une gestion simplifiée de vente à domicile.

Schéma relationnel :

CATALOGUE (référence, libellé, taille, couleur, prix, qté_en_stock)
CLIENT (code_client, nom, prénom, adresse, tél)
COMMANDE (no_commande, date, *code_client*)
ARTICLE_COMMANDE (no_commande, référence, qté)

Remarque : les clés primaires sont soulignées, les clés étrangères sont en italique.

Travail demandé :

Ecrire en SQL la création des tables suivantes :
CATALOGUE, CLIENT, COMMANDE, ARTICLE_COMMANDE.

Insérer des données cohérentes dans la base

Chapitre 2 : Oracle - Premières requêtes

2.1 - Requêtes sur une table

a) Syntaxe générale :

```
SELECT ...        -- tout (*) ou projection sur une ou plusieurs colonnes
  FROM ...        -- liste des tables
  WHERE expression -- conditions de jointures, restrictions
    GROUP BY ...  -- groupement des occurrences par paquets, les résultats concernent les paquets
HAVING ...        -- condition sur les "paquets"
  ORDER BY ...    -- rangement des lignes de résultats selon un critère
```

Remarque : le résultat de la sélection est une table (virtuelle).

b) Sélection simple : les clauses SELECT et FROM

- Contenu d'une table

  SELECT * FROM T ;

  Base exemple :
  *SELECT * FROM client*

- Projection sur certaines colonnes
  Rappel :
  Considérons une Relation R de n attributs A1, A2, ... An. L'ensemble des occurrences de R peut être considéré comme un sous-ensemble de l'ensemble produit D=D1xD2x...xDn, Di étant le domaine de valeurs associé à l'attribut Ai pour tout i.
  La projection de R sur un sous-ensemble des attributs est un sous-ensemble de D, le résultat ne contient donc pas de doublons (lignes identiques).
  Si dans la projection aucun composant de la clé n'est supprimé, les lignes obtenues en supprimant les attributs non désirés seront toutes distinctes, par contre si au moins un des composants de la clé est absent dans la projection,

il faut d'abord extraire des occurrences les colonnes retenues dans la projection, puis éliminer les lignes en double (les doublons).

SELECT [ALL|DISTINCT] c1, c2, ..., ck
FROM T;

Remarque concernant ORACLE : on peut intercaler ALL ou DISTINCT entre la commande SELECT et le nom des colonnes :
ALL (par défaut) permet d'afficher toutes les occurrences, même les doublons
DISTINCT n'affiche que les occurrences distinctes.

SELECT nom, prénom, tél
FROM client;

- Opérations de calcul (+, -, *, /) sur des colonnes
- -- référence, libellé et prix TTC des articles du catalogue
- *SELECT référence, libellé, prix*1.206 [AS] prixTTC*
- *FROM catalogue;*
-
- -- valeur HT du stock pour chaque article
- *SELECT référence, libellé, prix*qté_en_stock [AS] valeur_stock*
  FROM catalogue;

On remarquera :

- les lignes de commentaires (précédées de --)
- les alias de colonnes (AS est facultatif) : nouveau nom donné à la colonne résultat

c) Sélections avec qualification

- Restriction : utilisation de la clause **WHERE *expression***

L'expression est évaluée pour chaque tuple (ou n-uplet, occurrence, ligne) de la table.
Seuls les tuples pour lesquels l'expression est vraie sont retenus.

*SELECT **
FROM catalogue
WHERE qté_en_stock=0;

- Opérateurs SQL utilisés avec la clause WHERE

  - arithmétiques : **+ - * /**
  - sur les dates : **+ -**
  - sur les chaînes de caractères : **||** (concaténation)
  - de comparaison (nombres, chaînes ou dates) :
    =, != ou <>, <, <=, >, >=
    ordre lexicographique pour les chaînes de caractères
    ordre chronologique pour les dates
  - *SELECT no_commande, code_client*
  - *FROM commande*
    WHERE date_commande **>=** *'01-JAN-1999';*

- booléens : **AND, OR, NOT**

```
-- commandes d'un client donné (de code 1234) depuis le 1er janvier 1999
SELECT no_commande, code_client
   FROM commande
   WHERE code_client=1234
   AND date_commande >= '01-JAN-1999';
```

- autres opérateurs :

  1) x **BETWEEN** a **AND** b

  (expression vraie si x est compris entre a et b, bornes incluses)

```
-- numéros des commandes passées entre deux dates
SELECT no_commande
   FROM commande
   WHERE date_commande BETWEEN "01-JAN-1999" AND '31-JAN-1999';
```

2) x **IN** (v1, v2, ..., vk)

(expression vraie si x est égal à un des éléments de la liste) v1, v2, ..., vk doivent être de même type que x

```
-- numéros des commandes d'articles figurant dans une liste
SELECT no_commande
   FROM article_commande
   WHERE référence IN ('P201','P203','P206');
```

3) x **NOT IN** (v1, v2, ..., vk)

(expression vraie si x est différent de tous les éléments de la liste)

```
-- liste des articles qui sont ni rouges, ni verts, ni jaunes
SELECT référence, libellé, couleur, taille
   FROM catalogue
   WHERE couleur NOT IN ('rouge','vert','jaune');
```

4) x **LIKE** y

expression vraie si les deux chaînes sont égales en tenant compte des caractères jokers de la seconde chaîne

le caractère % remplace n'importe quelle chaîne de caractères, même la chaîne vide

le caractère _ remplace exactement un caractère.

```
-- liste des clients dont le nom est DUPONT, DUPOND ou DUPONS
SELECT prénom, nom
   FROM client
   WHERE nom LIKE 'DUPON_';

   Résultat : PRENOM      NOM
              Jérome      DUPONT
              Julien      DUPOND
              Monique     DUPONS

  -- liste des clients dont le nom ressemble à MO[N]ET
SELECT prénom, nom
   FROM client
   WHERE nom LIKE 'MO%NET%';

   Résultat : PRENOM      NOM
              Francette   MONET
              Claude      MONET
              Jean        MONNET
              Isidore     MONNETER
              Françou     MONETTE
```

5) x **NOT LIKE** y

x **IS NULL**
(expresion vraie si la valeur de la colonne n'est pas renseignée)
x **IS NOT NULL**
(expresion vraie si la valeur de la colonne est renseignée)

```
-- liste des prénom, nom, numéro de téléphone des clients
 SELECT prénom, nom, tél
     FROM client
     WHERE tél IS NOT NULL ;
-- liste des clients dont on n'a pas le numéro de téléphone
 SELECT prénom, nom
     FROM client
     WHERE tél IS NULL ;
```

6) x *op* **ANY** (...), x *op***ALL** (...), x *op***SOME** (...), x *op* **EXISTS** (...) avec (op : =, !=, <, <=, > ou >=)

ALL : la condition est vraie si elle est vérifiée pour toutes les occurrences qui interviennent dans la comparaison. Nous pouvons remarquer que dans le cas de l'opérateur numérique > cela revient à chercher un maximum.

Dans la base_exemple, trouver le produit le plus cher :

```
-- Occurrences insérées dans la table (auparavant vide) :
INSERT INTO catalogue VALUES('A000001','TEE-
SHIRT','XL','BLANC',150.00,5);
INSERT INTO catalogue VALUES('A000002','TEE-SHIRT','L','BLEU
NUIT',140.00,5);
INSERT INTO catalogue VALUES('A000003','TEE-
SHIRT','M','BLANC',130.00,6);
INSERT INTO catalogue VALUES('A000004','TEE-
SHIRT','XL','BLANC',170.00,3);
INSERT INTO catalogue VALUES('A000005','TEE-
SHIRT','XL','NOIR',160.00,9);

-- Requête (Attention ! Requête dans une requête (voir chapître 4)) :
SELECT libelle, prix
    FROM catalogue
    WHERE prix >= ALL(SELECT prix FROM catalogue);

-- Résultat de la requête :
LIBELLE             PRIX
----------- ----------
TEE-SHIRT           170
```

- Fonctions SQL (voir chapitre suivant)

2.2 Opérateurs de l'algèbre relationnelle

UNION [ALL], INTERSECT, MINUS

Opérateur UNION

Il permet d'obtenir un ensemble de lignes qui est la réunion des lignes obtenues par deux sélections.
Les lignes communes n'apparaitront qu'une seule fois, sauf si l'on a utilisé UNION ALL.
Les lignes de chaque sélection doivent être de même nature.
Les alias de colonnes n'apparaissent qu'une fois.
La clause ORDER BY est écrite en dernière ligne. (cette clause sera revue plus loin)

```
SELECT ...
  FROM ...
  WHERE ...
UNION
SELECT ...
  FROM ...
  WHERE ...
ORDER BY ... ;
```

Opérateur INTERSECT

Il permet d'obtenir l'ensemble des lignes communes à deux sélections

```
SELECT ...
INTERSECT
SELECT ... ;
```

Opérateur MINUS

Il permet de retirer d'une sélection les lignes présentes dans une deuxième sélection.

```
SELECT ...
MINUS
SELECT ... ;
```

2.3 Opérations de Jointure

Rappel : notion de jointure

Une jointure nous sert à faire des requêtes concernant des données stockées dans plusieurs tables, mais ayant un certain lien entre elles.
Notons que rechercher dans la base exemple la liste des clients ayant passé commande à une date donnée nécessite de rechercher dans les tables CLIENT et COMMANDE en utilisant le lien *no_client*.
La jointure entre deux tables est le résultat de l'action qui consiste à créer une table virtuelle constituée d'un sous-ensemble du produit cartésien des deux tables, constitué de la manière suivante :

- on considère le produit cartésien des deux tables, c'est à dire une table ayant pour colonnes toutes les colonnes des deux tables, et pour tuples les tuples formés de toutes les associations de chaque tuple d'une table avec chaque tuple de l'autre.
- on supprime ensuite tous les tuples incohérents, c'est à dire ceux qui ne vérifient pas la condition de jointure.

Cette condition de jointure est en rapport avec la construction du MCD et des contraintes liant les tables.

- enfin, on supprime les colonnes qui font double emploi.
- Remarque : à la jointure est souvent associée une projection sur certaines colonnes.

- **Jointure simple sur deux tables**

La requête est faite sur les deux tables : **FROM** table1, table2
La condition de jointure est spécifiée dans la clause **WHERE** *prédicat*
sous une forme du genre :
WHERE table1.col_k=table2.col_h
En l'absence de condition de jointure, le résultat obtenu est le produit cartésien des tables.

```
-- liste des clients qui ont passé commande depuis le 1er janvier 1999
SELECT DISTINCT prénom, nom
      FROM client,commande
      WHERE client.code_client=commande.code_client
      ANDdate_commande>'01-JAN-1999' ;
     -- noter la présence de DISTINCT
     -- jointure sur la colonne commune code_client
     -- il est nécessaire d'indiquer nomtable.nomcolonne en cas d'homonymes


-- Pour simplifier l'écriture on peut définir des alias de table, par exemple :
SELECT DISTINCT prénom, nom
       FROM client cl,commande co
       WHERE cl.code_client=co.code_client
       AND date_commande>'01-JAN-1999' ;
```

Equi-jointure : la condition de jointure est l'égalité entre une colonne de la première table et une colonne de la deuxième table.

- **Jointure sur plusieurs tables**

On indique la liste des tables sur lesquelles est faite la requête.
On écrit une condition complexe comprenant les conditions de jointure des tables prises deux à deux.

- **Jointure d'une table avec elle-même**

On peut avoir besoin de rassembler ou comparer des informations provenant de lignes différentes de la même table.
Dans ce cas il faut utiliser des alias de table
FROM table1 **Ta**, table1 **Tb**

- **Jointure externe**

Lorsqu'une ligne d'une table figurant dans une jointure ne peut être liée à aucune ligne de la seconde table parce qu'elle ne satisfait pas à la condition d'équi-jointure entre les deux tables, elle ne figure pas dans le résultat de la jointure.
En Oracle, une option permet de faire figurer dans le résultat les lignes satisfaisant à la condition d'équi-jointure plus celles n'ayant pas de correspondant. Cette option s'obtient en accolant (+) au nom de colonne de la table dans laquelle manquent les éléments, dans la condition d'équi-jointure.

2.4 Rangement des résultats

Clause ORDER BY
Il s'agit d'ordonner les résultats ; dans une requête, cette clause est donc toujours placée en dernier, après toutes les autres clauses.

On peut ordonner les occurrences obtenues dans la sélection selon les valeurs d'une colonne nommée par son alias (ou à défaut son nom ou l'expression qui la définit), par son numéro d'ordre au niveau de l'affichage ou même selon une partie de la colonne (sous_chaîne).
Le sens du rangement est indiqué derrière chaque indicatif de rangement :

ASC : ordre croissant du rangement (par défaut)
DESC : ordre décroissant

```
SELECT ...
  FROM ...
  WHERE ...
  ORDER BY colonne [ASC|DESC];
  (ASC est facultatif (valeur par défaut))
```

Pour ordonner selon une liste de critères :
- les critères sont donnés dans l'ordre d'importance : du plus important au moins important
- derrière chaque critère est indiqué le sens du rangement : ASC (facultatif) ou DESC
- exemple : ORDER BY col1 DESC, col2 ASC

EXEMPLES :

```
-- liste des clients dont on n'a pas le numéro de téléphone
-- rangés par ordre alphabétique des noms
SELECT prénom, nom
    FROM client
    WHERE tél IS NULL
    ORDER BY nom ;


-- liste des clients dont on n'a pas le numéro de téléphone
-- rangés par ordre alphabétique des noms et des prénoms
SELECT prénom, nom
    FROM client
    WHERE tél IS NULL
    ORDER BY nom, prénom ;
   -- ordre demandé par le nom des colonnes
  ou
SELECT prénom, nom
    FROM client
    WHERE tél IS NULL
    ORDER BY 2, 1 ;
   -- ordre demandé par le numéro de sortie des colonnes
```

-- ordre croissant sur la date :

-- on peut transformer la date en chaîne de caractères et extraire les sous-chaînes qui nous intéressent

```
BREAK ON code_client ON nom
SELECT nom, TO_CHAR(date_commande, 'dd/mm/yyyy') dates
  FROM E_client cl, E_commande co
  WHERE cl.code_client = co.code_client
  ORDER BY nom, SUBSTR(dates,7,4), SUBSTR(dates,4,2), SUBSTR(dates,1,2) ;
```

2.5 Présentation des résultats

FORMATAGE DES RESULTATS

1. FORMATAGE D'UNE PAGE

Longueur de page, de ligne :
SET PAGESIZE n -- longueur de page
SET LINESIZE n -- longueur de ligne
SET HEADING ON -- affichage des titres des colonnes
SET HEADING OFF -- suppression des titres des colonnes

Entete de page :

TTITLE [COL n|SKIP n|TAB n|LEFT|RIGHT|CENTER| 'texte']

TAB n : mettre le texte apres n tabulations

COL n : mettre le texte a partir de la colonne n

SKIP n : sauter n lignes

TTITLE OFF : pour annuler l'affichage du titre

TTITLE ON : pour retablir l'affichage du titre

BTITLE : bas de page (mêmes commandes)

2. FORMATAGE DE COLONNES

COLUMN nom_colonne options

Les options s'appliquent à toutes les colonnes de nom (ou d'alias) nom_colonne pendant la durée d'une session ou jusqu'à ce que d'autres options les remplacent.

- Consultation des options :

  COL nom_colonne

- Principales options :

  HEADING chaine : change l'entête de la colonne.

  Mettre des apostrophes si la chaîne contient des blancs
  Le caractère "|" force le retour ligne

  FORMAT masque

  Chaînes de caractères :
  - FORMAT An => chaine de n caractères

  Nombres :

- FORMAT 99999 => taille d'affichage du nombre
- FORMAT 09999 => complète le nombre par des 0 à gauche
- FORMAT $999 => préfixe le nombre avec $
- FORMAT 999.99 => aligne le point décimal dans la position indiquée

JUSTIFY L | R | C
WRAPPED | TRUNCATED
NEWLINE : passage a la ligne avant l'affichage de la colonne
CLEAR : toutes les options sont annulees
ON/OFF : bascule ``options définies"/"options par defaut"

3. RUPTURE DE SEQUENCE

BREAK ON element_de_rupture [action] [ON element_de_rupture [action]]

où element_de_rupture =

un nom de colonne ou une expression (la valeur n'est pas répétée si elle est identique à celle de l'occurrence précédente))
RAW : l'action demandée est executée à chaque ligne
PAGE : l'action demandée est executée à chaque page
REPORT : l'action demandée est executée en fin de rapport (report).

et où action =

SKIP n : saut de n lignes
SKIP PAGE : commencer une nouvelle page

4. TOTALISATION

COMPUTE operation OF col1 col2, ... ON element_de_rupture;

Opérations possibles : SUM, MIN, MAX, AVG, STD (écart-type), VAR (variance), COUNT (nombre de valeurs non nulles de la colonne), NUM (nombre de valeurs de la colonne y compris les valeurs nulles).

5. Effacement de commandes de formatage

Les commandes sont valables pendant toute la session, tant qu'elles ne sont pas remplacées par d'autres ou effacées.

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES

6. Envoi des résultats d'une commande dans un fichier

SPOOL nom_fichier
Séquence de commandes SQL
SPOOL OFF (pour arreter)

2.6 Exercices

FEUILLE D'EXERCICES SQL, SQL*PLUS N°02

VOCABULAIRE : BASES DE DONNEES => ORACLE

Relation => table :

. Nom de la Relation => Nom de la Table

. Attribut => titre d'une colonne

. Valeur d'un attribut => valeur d'un élément de la colonne

. Domaine d'un attribut => valeurs possibles dans la colonne

. Contrainte sur les valeurs d'un attribut => contrainte de colonne

. Occurrence (tuple, n-uplet) => ligne (nombre variable de lignes)

. Clé primaire composée d'un seul attribut => contrainte de colonne

. Clé primaire composée de plusieurs attributs => contrainte de table

. Clé étrangère => contrainte de colonne

RAPPELS

Projection : SELECT c1,c2,...ck FROM T ;

Restriction : SELECT * FROM T WHERE cond ;

Jointure : SELECT .... FROM T1, T2 WHERE T1.c1 = T2.c2 ;

(ou avec des alias de tables :)

SELECT ... FROM T1 a, T2 b

WHERE a.c1 = b.c2 ;

Remarque : SELECT [ALL|DISTINCT] pour spécifier si l'on veut toutes les occurrences ou seulement celles qui sont distinctes. (par défaut ALL)

REQUETES :

1)

Requête sur une seule table : projection sur trois colonnes

restriction à pays = 'FRANCE'

Présentation des résultats : alias de colonne

ORDER BY : tri croissant sur une colonne

Afficher la liste des artistes nés en FRANCE par ordre croissant des noms.

Titre des colonnes de résultats : NOM LOCALITE DATE_NAISSANCE

2)

Requête sur une seule table : projection sur trois colonnes,
restriction à une colonne contenant la chaîne 'FEMME'
LIKE, %

Présentation des résultats : tri croissant sur deux colonnes.

Afficher la liste des oeuvres dont le titre contient le mot FEMME.
Tri par valeur, titre.
Titre des colonnes de résultats : DATE_CREATION TITRE VALEUR

3)

Requête sur une seule table : restriction à un intervalle de valeurs,
OU à des valeurs non définies
ET restriction à un champ contenant
un OU plusieurs mots donnés.
-- OR, BETWEEN, IS NULL

Présentation des résultats : des colonnes de largeur prédéfinie
tri croissant sur deux colonnes.

Afficher la liste des oeuvres dont la valeur est comprise entre 380 000
et 950 000 F ou bien n'est pas définie et dont le titre contient le mot FEMME
ou le mot DEMOISELLE ou le mot ENFANT
Tri par valeur, titre.
Titre des colonnes de résultats : TITRE DATE_CREATION VALEUR

4)

Requête sur une seule table : projection sur 4 colonnes, alias de colonne
Restriction avec AND, OR, LIKE,
NOT LIKE, NOT IN, IS NOT NULL

Présentation des résultats : tri sur 3 colonnes,
Deux colonnes non répétées
Longueurs de lignes, de page

Afficher la liste des artistes vivants, nés en FRANCE ailleurs qu'à PARIS
ou nés dans un pays autre que la FRANCE et l'ALLEMAGNE.
Tri par pays, localité, nom.
Titre des colonnes de résultats : PAYS LOCALITE NOM DATE_NAISSANCE

5)

Requête sur deux tables : jointure naturelle sur une colonne, alias de table

Présentation des résultats : rupture de séquence

Afficher la liste des oeuvres des artistes nés en ESPAGNE.

Tri par artiste, année de création

Le nom de l'artiste ne devra pas être répété.

Titre des colonnes de résultats : NOM_ARTISTE DATE_CREATION TITRE_OEUVRE

6)

Requête sur deux tables : jointure naturelle sur 1 colonne, alias de table

Présentation des résultats : tri et rupture de séquence

Afficher la liste des oeuvres de tous les artistes nés hors de FRANCE.

Tri par nom, année de création.

Pays et nom_artiste ne devront pas être répétés.

Titre des colonnes de résultats :

PAYS NOM_ARTISTE DATE_CREATION TITRE

7)

-- alias de table obligatoire

Afficher la liste des artistes dont le maître est répertorié.

Tri par nom.

Titre des colonnes de résultats :

NOM_ARTISTE NAISSANCE_ARTISTE NOM_MAITRE NAISSANCE_MAITRE

8)

-- jointure sur plusieurs tables

-- Rupture de séquence, une occurrence par ligne (tronquer le titre si besoin)

Pour un artiste donné, obtenir la liste des oeuvres exposées dans chaque musée

Ordonner les résultats selon le nom du musée, ne pas répéter le nom du musée.

Titre des colonnes de résultats :

NOM_MUSEE DEBUT_EXPO FIN_EXPO TITRE_OEUVRE

9)

-- table utilisée dans la requête et non dans l'affichage

-- jointures multiples, rangement des résultats, rupture

Obtenir la liste des artistes ayant exposé dans la même exposition.
On pourra entrer le nom du musée en paramètre ou faire la requête
pour tous les musées.
Titre des colonnes de résultats :
NOM_MUSEE DEBUT_EXPO FIN_EXPO NOM_ARTISTE

10)
-- jointures
-- rangement alphabétique par rapport à une colonne
-- rupture sur le nom de chaque artiste avec saut de ligne
Obtenir la liste des techniques utilisées par chaque artiste Espagnol
Rangement par ordre alphabétique des noms des artistes
Saut de ligne entre chaque artiste.
Titre des colonnes de résultats :
NOM_ARTISTE LIBELLE_TECHNIQUE

11)
-- Double rupture
Obtenir la liste des artistes franlçais ou espagnols pour chaque technique
utilisée.
Ordonner les résultats selon la technique, le pays, l'artiste.
Titre des colonnes de résultats :
LIBELLE_TECHNIQUE PAYS NOM_ARTISTE

12)
-- Elimination de doublons prévisibles dans le résultat
Obtenir la liste des artistes qui ont travaillé sur chacun des supports
répertoriés.
Ordonner les résultats selon le support puis le nom de l'artiste.
Attention ! On peut retrouver le même artiste pour plusieurs supports,
mais surtout on peut retrouver plusieurs fois le même artiste pour le même
support (autant de fois que d'oeuvres de l'artiste sur ce support).

Chapitre 3 : Oracle - Fonctions

3.1 - Fonctions simples

- **numériques**
  ABS (n), COS(n), SIN(n), EXP(n), LN(n), SQRT(n)
  CEIL (n) : le plus petit entier >= n
  FLOOR (n) : le plus grand entier <= n
  MOD (m,n) : le reste de la division entière de m par n
  POWER (m,n) : m puissance n
  LOG (m,n) : logarithme en base m de n
  ROUND (n,m) : la valeur de n arrondie à m positions à droite si m est positif, à gauche sinon
  TRUNC (n,m) : tronque au lieu d'arrondir
  SIGN (n) : -1 si n < 0, 0 si n = 0, +1 si n > 0

- **caractères**
  CHR (n) : caractère de code ASCII n
  ASCII (car) : le code ascii de car (ou du premier caractère de car)
  LOWER (ch) : met la chaîne ch en minuscules
  UPPER (ch) : met la chaîne ch en majuscules
  LENGTH (ch) : longueur de la chaîne ch
  SUBSTR (ch,pos,[long]) : extrait de la chaîne ch une sous-chaîne de longueur long commençant en position pos.
  Si pos est négatif la position est comptée à partir de la droite de ch.
  RTRIM (ch,' ') : enlève les espaces à droite de la chaîne, etc.

- **dates**
  SYSDATE : date et heure courantes du système
  GREATEST (d1,d2) : la date la plus récente
  LEAST (d1,d2) : la date la plus ancienne
  MONTH-BETWEEN (d1,d2) : nombre de mois entre 2 dates

- **conversions**
  de date vers chaîne : TO_CHAR (date, format)
  de chaîne vers date : TO_DATE (chaîne, format)
  de chaîne vers nombre : TO_NUMBER (chaîne, format)

  On trouve comme format de date (avec un séparateur au choix), par exemple :

  'dd-mon-yyyy' => 26-NOV-1999
  'dd month yyyy' => 26 NOVEMBER 1999
  'dd/mm/yy' => 26/11/99
  'dd/mm/yyyy' => 26/11/1999

- **autres fonctions**
  NVL (exp1, exp2) : retourne exp2 si exp1 est NULL, exp1 sinon.

  Par exemple, on peut s'en servir pour remplacer NULL par des étoiles à l'affichage
  ```
  SELECT NVL (nom_col,'*****') ...
  ```
  ou pour éliminer les occurrences ayant 0 et garder celles qui ont NULL dans une colonne de type numérique en utilisant la condition :
  ```
  WHERE NVL (nom_col,1) <> 0
  ```

DECODE (expression, valeur1, résultat1, valeur2, résultat2 ...) : retourne
résultat1 si expression = valeur1
résultat2 si expression = valeur2, etc.

Par exemple, on peut s'en servir pour exprimer un résultat dépendant d'un seuil de température

```
SELECT DECODE (SIGN(temperature-seuil), -1, résultat1, 0, résultat2, 1, résultat2)
    FROM nom_table
   WHERE ... ;
```

3.2 - Fonctions d'Agrégat

Une fonction agrégat est une fonction qui retourne une valeur calculée sur un groupe de lignes :

```
SELECT f(...), ...
  FROM T ...
  [WHERE ...]
  [GROUP BY ...]
```

Les valeurs nulles (NULL) sont ignorées sauf pour la fonction COUNT.

<table>
<tr><td rowspan="5">COUNT (-)</td><td colspan="2">nombre de lignes intervenant dans le résultat de la requête</td></tr>
<tr><td>COUNT(*)</td><td>calcule le nombre d'occurrences</td></tr>
<tr><td>COUNT(col)</td><td>calcule le nombre de valeurs non nulles de la colonne</td></tr>
<tr><td>COUNT (DISTINCT exp)</td><td>le calcul n'est fait que sur l'ensemble des valeurs distinctes de l'expression</td></tr>
<tr><td colspan="2">SELECT COUNT (DISTINCT code_client) nb_de_clients
FROM commande;</td></tr>
<tr><td rowspan="2">SUM (-)</td><td colspan="2">somme des valeurs retenues dans la requête</td></tr>
<tr><td colspan="2">SELECT SUM (prix*qté_en_stock) Montant_HT
FROM commande c, article_commande ac
WHERE c.no_commande=1234
AND c.no_commande=ac.no_commande;</td></tr>
<tr><td>AVG (-)</td><td colspan="2">moyenne des valeurs retenues dans la requête</td></tr>
</table>

	SELECT AVG (qté) qté_moyenne *FROM article_commande* *WHERE référence="A145T38"*
MAX (-)	la plus grande valeur de l'expression demandée
	SELECT MAX (qté) plus_grosse_commande *FROM article_commande* *WHERE référence="A145T38"*
MIN (-)	la plus petite valeur de l'expression demandée
VARIANCE (-)	variance calculée sur les valeurs retenues dans la requête
STDDEV (-)	écart-type calculé sur les valeurs retenues dans la requête

3.3 - Groupement d'occurrences

- **Clause GROUP BY**
  Chaque occurrence obtenue est le résultat d'un regroupement par paquets de lignes selon la valeur de l'expression donnée dans la clause GROUP BY.

  SELECT ...
  GROUP BY e1, e2, ... ei, ..., en
  où ei désigne une colonne ou une expression,
  les alias ne sont pas acceptés.

  SELECT code_client, COUNT (no_commande) nombre_de_commandes
  FROM commande
  GROUP BY code_client;

  La liste des éléments sélectionnés (sur la ligne SELECT ...) doit être cohérente avec les expressions de la clause GROUP BY.
  Elle peut contenir :

  - des fonctions d'agrégat ; l'agrégation se fera selon le groupement demandé

    Exemple : sélection des n° de commandes et de leur montant.
    *SELECT no_commande, SUM (prix*qté)*
    FROM article_commande ac, catalogue c
    WHERE ac.reference = c.reference
    ***GROUP BY** ac.no_commande;*

  - des expressions existant dans le GROUP BY
  - des constantes

- des expressions cohérentes avec la demande d'agrégation
  (ex. : une colonne ayant plusieurs valeurs distinctes dans un même groupe ne peut pas exister dans la ligne SELECT ...

Remarque : Le résultat de la sélection est constitué de l'ensemble des groupes, autrement dit, chaque occurrence obtenue est le résultat du traitement d'un groupe de lignes.

- **Clause HAVING**
  Exécution de la clause GROUP BY sur des paquets de lignes, puis restriction aux occurrences obtenues satisfaisant à une condition C.
  La clause HAVING n'accepte pas les alias de colonne.

  SELECT ...
  GROUP BY ...
  HAVING C

  Par exemple, sélection des numéros de commande et de leur montant pour toutes les commandes passées en 1999.

```
SELECT co.no_commande, SUM (prix*qté) montant
  FROM commande co, article_commande ac, catalogue ca
  WHERE ac.reference = ca.reference
  AND ac.no_commande = co.no_commande
  GROUP BY co.no_commande, co.date_commande
  HAVING co.date_commande >= '01-01-1999' AND co.date_commande <= '31-12-1999';
```

  Par exemple, sélection des clients ayant fait au moins 3 commandes en 5 ans.

```
SELECT code_client, count (no_commande)
  FROM commande
  WHERE TO_NUMBER (TO_CHAR (date_commande,'yyyy'),9999) >
TO_NUMBER(TO_CHAR(sysdate,'yyyy'),9999)-5
  GROUP BY code_client
  HAVING count (no_commande) >= 3;
```

3.4 - Présentation : les ruptures de séquences

3. RUPTURE DE SEQUENCE

BREAK ON element_de_rupture [action] [ON element_de_rupture [action]]

où element_de_rupture =

un nom de colonne ou une expression (la valeur n'est pas répétée si elle est identique à celle de l'occurrence précédente))
RAW : l'action demandée est executée à chaque ligne
PAGE : l'action demandée est executée à chaque page
REPORT : l'action demandée est executée en fin de rapport (report).

et où action =

SKIP n : saut de n lignes
SKIP PAGE : commencer une nouvelle page

4. TOTALISATION

COMPUTE operation OF col1 col2, ... ON element_de_rupture;

Opérations possibles : SUM, MIN, MAX, AVG, STD (écart-type), VAR (variance), COUNT (nombre de valeurs non nulles de la colonne), NUM (nombre de valeurs de la colonne y compris les valeurs nulles).

5. Effacement de commandes de formatage

Les commandes sont valables pendant toute la session, tant qu'elles ne sont pas remplacées par d'autres ou effacées.

CLEAR COLUMNS
CLEAR BREAKS
CLEAR COMPUTES

6. Envoi des résultats d'une commande dans un fichier

SPOOL nom_fichier
Séquence de commandes SQL
SPOOL OFF (pour arreter)

- ## 3.5 - Exercices

FEUILLE D'EXERCICES SQL, SQL*PLUS N°03

I. Fonctions
==========

1)
-- Extraction de l'année dans une date
Liste des artistes nés en FRANCE
Tri par ordre croissant des années de naissance et des noms.

ANNEE_NAISSANCE NOM LOCALITE

2)
-- fonctions d'agrégat
Afficher le nombre d'artistes ayant des oeuvres et le nombre d'oeuvres répertoriées
dans la base

NOMBRE-ARTISTES NOMBRE_OEUVRES

3)
-- fonctions d'agrégat, groupement des occurrences par paquets
Afficher pour chaque artiste le nombre d'oeuvres et la valeur moyennedes
oeuvres de l'artiste.

Rangement par ordre alphabétique des noms d'artistes.

NOM NB_OEUVRES VALEUR_MOYENNE

4)

-- fonctions d'agrégat, groupement des occurrences par paquets

-- tri décroissant

Afficher pour chaque pays le nombre d'oeuvres répertoriées, la valeur globale de ces oeuvres

Rangement décroissant par valeur globale, croissant par nombre d'oeuvres et par pays.

PAYS NB_OEUVRES VALEUR_GLOBALE

5)

-- Fonction ayant deux actions selon que la valeur existe ou non.

-- Rangement selon une sous-chaîne extraite d'une colonne.

Liste des noms, date de naissance et de décès des artistes français. Des étoiles remplaceront la date de décès si celle-ci n'existe pas.

Tri par date de naissance et par nom

NOM DATE_NAISSANCE DATE_DECES

6)

-- travail sur les dates.

-- Elément de rupture : l'année de naissance.

Liste des artistes nés dans la même décennie que PICASSO.

Tri croissant selon l'année de naissance.

ANNEE_NAISSANCE NOM LOCALITE PAYS

7)

-- traitements par lots d'occurrences et conservation d'une partie des résultats seulement.

Liste des artistes, nombre d'oeuvres et valeur moyenne des oeuvres pour les artistes dont le nombre d'oeuvres est supérieur à un nombre donné.

tri décroissant selon la valeur moyenne et par ordre alphabétique des noms.

NOM_ARTISTE NB_OEUVRES VALEUR_MOYENNE

8)
-- jointure externe
-- remplacement de null par 'non connu'
-- tri selon un critère calculé
Liste des artistes, nombre d'oeuvres et valeur moyenne des
oeuvres pour tous les artistes, même ceux qui n'ont pas d'oeuvre.
Tri décroissant selon la valeur moyenne et par ordre alphabétique des noms.
NOM_ARTISTE NB_OEUVRES VALEUR_MOYENNE

9)
-- group by
-- having
Nombre d'oeuvres répertoriées, nombre de supports pour les artistes
français, allemands, russes et espagnols.
Rupture au niveau de la nationalité de l'artiste avec saut de ligne.
Tri alphabétique du pays, et pour chacun, tri décroissant selon le
nombre d'oeuvres.
PAYS ARTISTE NB_OEUVRES NB_SUPPORTS

10)
-- question pouvant être comprise et traitée de plusieurs manières
Trouver l'oeuvre la plus chère, l'oeuvre la moins chère
Affichage selon le traitement fait.
On peut afficher : TITRE_ONEREUX VALEUR_MAX TITRE_ABORDABLE VALEUR_MIN

==========

II. Jointures, ruptures

11) Liste des oeuvres exposées dans un musée français.
Caractéristiques musée, artiste non répétées.
Cumul des valeurs par musée
MUSEE LOCALITE PAYS ARTISTE OEUVRE

III. Divers

12)
-- utilisation de la différence de dates
-- un même artiste apparait autant de fois qu'il a créé d'oeuvres avant un âge donné.

Liste des artistes ayant créé des oeuvres avant un âge passé en paramètre.
Attention la date de création est une année (chaîne de longueur 4).
NOM DATE_NAISSANCE LOCALITE PAYS

13)
-- utilisation de la fonction DECODE pour donner le sexe (HOMME, FEMME)
-- un artiste peut avoir plusieurs tendances.

Liste des tendances de tous les artistes nés hors de FRANCE.
Caractéristiques d'un artiste non répétées.
Tri croissant par nom, tendance.
NOM SEXE DATE_NAISSANCE TENDANCE

Chapitre 4 : Oracle - Sous-requêtes, Vues

4.1 - Sous-Requêtes

C'est une requête incluse dans une expression SQL.
Par exemple, une expression utilisée dans la clause WHERE peut être le résultat d'une requête qui est alors appelée sous-requête.
Par exemple, liste des prénom, nom et numéro de téléphone des clients qui ont commandé tel article.

```
SELECT prenom, nom, tel
  FROM client
  WHERE code_client IN
    (SELECT code_client
      FROM commande c, article_commande ac
      WHERE c.no_commande=ac.no_commande
      AND ac.reference='A102T44');
```

- La sous-requête ramène une seule valeur. Dans ce cas, on peut utiliser

WHERE exp1 *op* (SELECT exp2 ...)
où op est un opérateur de comparaison.
exp1 et exp2 doivent être cohérents.

Par exemple : client ayant le plus grand numéro de commande.

```
SELECT code-client, nom, no_commande
  FROM client cl, commande co
  WHERE cl.code_client = co.code_client
  AND no_commande =
    SELECT MAX(no_commande)
    FROM commande;
```

- La sous-requête peut ramener plusieurs occurrences d'une expression. Elle est utilisée dans :

WHERE exp1 [NOT] IN
(SELECT exp2 ...)

Par exemple : tous les numéros de commandes d'un client ayant commandé un article de référence X;

```
BREAK ON code_client on nom
SELECT code-client, nom, no_commande, date_commande
  FROM client cl, commande co
  WHERE cl.code_client = co.code_client
  AND co.no_commande IN
    (SELECT no_commande
    FROM article_commande;
    WHERE code_article = X)
    ORDER BY nom, date_commande;
```

WHERE exp1 *op_de_comparaison* {ALL|ANY|SOME}
(SELECT exp2 ...)

- ALL : la condition est vraie si la comparaison est vraie pour chacune des valeurs retournées.
- ANY : la condition est vraie si la comparaison est vraie pour au moins une des valeurs retournées.
- SOME : synonyme de ANY

```
SELECT no_commande
  FROM commande
  WHERE no_commande >= ALL
    (SELECT no_commande
      FROM commande);
{revient à prendre le max}
```

- La sous-requête peut retourner une ou plusieurs occurrences d'une liste d'éléments.

WHERE (exp1, exp2, ...) [NOT] IN
(SELECT ...)
La liste des éléments retournés par la sous-requête doit être du même type que (exp1, exp2, ...)

```
SELECT code_article, libelle, prix*qte montant
  FROM article_commande ac, catalogue ca
  WHERE ac.code_article=ca.code_article
  AND (ac.code_article, qte) IN
    (SELECT code_article, qte
      FROM article_commande
       WHERE no_commande IN
        (SELECT no_commande
          FROM commande
          WHERE date_commande>'01-JAN-2000'));
```

- Sous-requête synchronisée avec la requête principale
La sous-requête utilise une expression contenant au moins une colonne d'une table T de la requête principale.
Il est alors nécessaire de définir un alias de la table T.

Exemple :
A partir d'une commande C d'un client recherche de toutes les commandes dans lesquelles il a déjà commandé un même article.

```
SELECT no_commande, reference, qté
  FROM article_commande AC, commande co
  WHERE ac.no_commande=co.no_commande
  AND ac.reference IN
    (SELECT reference
      FROM article_commande, commande
      WHERE article_commande.no_commande=commande.no_commande
      AND commande.no_client=co.no_client
      AND commande.no__commande=C);
```

- Autre sous-requête **WHERE [NOT] EXISTS (SELECT ...)** :
  renvoie un booléen Vrai ou Faux selon le résultat de la requête
  - Vrai si le résultat produit au moins une ligne
  - Faux sinon.

Exemple :

```
-- les noms des clients qui ont commandé un produit de
-- référence donnée.

SELECT prenom, nom
  FROM client cl, commande co
  WHERE cl.code_client = co.code_client
  AND EXISTS (SELECT *
    FROM article_commande ac
    WHERE reference = 'AR92L'
    AND ac.no_commande = co.no_commande);
```

4.2 - Utilisation des requêtes dans d'autres commandes SQL

Up

- Insertion dans une table des occurrences obtenues comme résultat d'une requête.

Exemple :

```
-- création d'une table temporaire
CREATE TABLE temp
    (code_client VARCHAR2(10),
     nb_commandes NUMBER(3));

-- puis insertion de données.
INSERT INTO temp
  (SELECT code_client, count(*)
    FROM commande
    GROUP BY code_client);
```

- Mise à jour d'une table à partir d'une requête

Exemple :

```
-- Création d'une table non nécessaire dans la Base de Données
    -- mais utile à un moment donné, elle pourra être détruite plus tard.
        CREATE TABLE achats AS
  SELECT prenom, nom, no_commande, date_commande
  FROM commande co, client cl
  WHERE co.code_client=cl.code_client;
```

```
-- ajout d'une colonne dans la table achats contenant le montant de la
commande
    ALTER TABLE achats
      ADD (montant NUMBER(10,2));

-- insertion des montants dans la table achats
-- synchronisation entre la table mise à jour et les occurrences extraites dans
la sélection
    UPDATE achats a
    SET montant = (SELECT SUM(qté*prix)
                   FROM article_commande ac, catalogue c
                   WHERE ac.référence=c.référence
                   AND ac.no_commande=a.no_commande)
    WHERE a.no_commande = ac.no_commande ;
```

4.3 - Vues Up

On peut stocker une requête dans une Vue.
Une vue a un nom et est définie par une requête.
Elle est stokée (nom et définition) dans le dictionnaire de données de la base.

Une vue se comporte comme une table contenant le résultat d'une requête ; elle peut être utilisée comme une table
pour consulter (et sous certaines conditions modifier) les données de la base.
Lorsqu'une requête fait appel à une vue, le SGBD substitue à la vue sa définition et c'est cette dernière qui sera exécutée.

- Création d'une vue

```
CREATE [OR REPLACE] VIEW nom_vue (Col1, Col2, ...)

    AS SELECT ... ;
     -- Noms de colonnes qui seront associés en nombre et ordre
     -- aux éléments du SELECT
```

ou

```
CREATE [OR REPLACE] VIEW nom_vue (Col1:Type1, Col2:Type2, ...)
  AS SELECT ... ;
   -- Schéma de la vue qui doit être cohérent avec
   -- les éléments du SELECT
```

ou

CREATE [OR REPLACE] VIEW nom_vue
AS SELECT ... ; -- la vue reprend les noms des colonnes ou alias de la requête

Attention ! On ne peut pas utiliser de clause ORDER BY.

- Renommage d'une vue

RENAME ancien_nom nouveau_nom ;

- Suppression d'une vue

DROP VIEW nom_vue

- Requêtes

Les requêtes sont identiques à celles qui sont faites sur les tables.
Une requête peut utiliser à la fois des tables et des vues.

- Mises à jour
- La mise à jour de tables de la base, tables utilisées par une vue, implique une mise à jour simultanée des données de la vue (évident puisque la vue ne stocke aucune donnée).
- La mise à jour directe d'une vue (et donc de la base à travers la vue) n'est en général pas possible sauf si la définition de la vue ne touche qu'une table et qu'elle n'a pas d'expression contenant des calculs, des fonctions d'agrégat.

- Exemple de création et d'utilisation d'une vue

Vue contenant les commandes d'un client :

```
CREATE VIEW client_X AS
   SELECT *
     FROM commande
     WHERE code_client=1345 ;
```

Recherche de la date de la dernière commande d'un client :

```
   SELECT MAX(date_commande)
     FROM client_X ;
```

4.4 - Utilisation des vues

Up

Une vue peut être utilisée pour :

- Restreindre l'accès à certaines lignes ou colonnes d'une table (sécurité des données)
- Simplifier la tâche d'un utilisateur final en lui fournissant une vue, résultat d'une requête complexe sur plusieurs tables.
- Ecrire des requêtes complexes dans lesquelles les vues jouent le rôle d'étapes intermédiaires.
- Sauvegarder des requêtes.

- Exemple : Restriction des données pour un représentant

  Vue contenant les clients d'un département :

  ```
  CREATE VIEW client_49 AS
     SELECT *
       FROM client
       WHERE code_postal LIKE '49%' ;
  ```

  De même : vue contenant les commandes des clients précédents.

  ```
  CREATE VIEW commande_49 AS
     SELECT *
       FROM commande co,client_49 cl
       WHERE co.code_client=cl.code_client
  ```

  On remarque l'utilisation de la vue client_49 pour définir une nouvelle vue.

FEUILLE D'EXERCICES SQL, SQL*PLUS N°04

1)

-- utilisation de la fonction DECODE pour donner le sexe (HOMME, FEMME)

-- sous requête pour les artistes nés hors de France,d'Espagne et d'Allemagne

-- un artiste peut avoir plusieurs tendances.

Liste des tendances de tous les artistes nés hors de FRANCE, d'ESPAGNE, d'ALLEMAGNE.

Caractéristiques d'un artiste non répétées.

Tri croissant par nom, tendance.

NOM SEXE DATE_NAISSANCE TENDANCE

2)

-- décomposition d'une date en années et mois

-- il peut ne pas y avoir de date de fin d'exposition

Déterminer la durée d'exposition (en années et mois) pour chaque oeuvre

qui a été ou qui est exposée dans un musée français.

Tri alphabétique par musée et pour chacun par ordre décroissant de la

durée d'exposition.

Faire une rupture de séquence avec saut de ligne pour chaque musée.

MUSEE OEUVRE DATE_DEBUT DATE_FIN DUREE

3)

-- création d'une vue permettant d'avoir le nombre d'oeuvres par artiste
-- utilisation de la vue pour déterminer une plus grande valeur.

Artiste ayant le plus grand nombre d'oeuvres répertoriées.
NOM DATE_NAISSANCE LOCALITE PAYS NB_OEUVRES

4)
-- rangement selon une valeur calculée
-- entrée au clavier avec &nom_paramètre (un plus de Sql*Plus)

Valeur moyenne des oeuvres par artiste s'exprimant selon une tendance
donnée. Tri par ordre décroissant de la valeur moyenne.
ARTISTE DATE_NAISSANCE LOCALITE PAYS VALEUR_MOYENNE

4.5 - Exercices

5)
-- vue sur les oeuvres de KLIMT
-- recherche d'une valeur maximale.
Musée(s) détenant le plus grand nombre d'oeuvres du peintre KLIMT GUSTAV
Tri décroissant par nombre d'oeuvres, alphabétique par musée.
MUSEE VILLE_MUSEE PAYS NB_OEUVRES

6)
-- vue permettant ensuite de déterminer ensuite une valeur moyenne
-- utilisation d'une valeur moyenne calculée sur une vue.
-- double rupture.

Musées dont la somme des valeurs des oeuvres est supérieure à la moyenne de
ces sommes de valeurs.
Pays et localité non répétés.
Tri croissant par pays, localite, décroissant pour la somme.
PAYS VILLE MUSEE MONTANT_GLOBAL

Chapitre 5 : PL/SQL : Premières notions

5.1 - Introduction

• PL/SQL : Langage de programmation
- utilisant les commandes SQL
- disposant d'un ensemble de fonctionnalités supplémentaires (programmation structurée)
• Déclarations
variables
variables initialisées
constantes
• Saisie au clavier par &valeur
• Structures algorithmiques (itérations, alternatives)

5.2 - Structure d'un bloc PL/SQL

Up

5.2.1 Schéma d'un script intégrant un bloc PL/SQL

```
- - bloc de commandes SQL

        _____ ;
        _____ ;
- - fin commandes SQL

- - Bloc PL/SQL

        DECLARE
            - - déclaration de variables et de curseurs
            _____ ;
            _____ ;
        BEGIN
            - - bloc d'instructions
            _____ ;
            _____ ;
        [EXCEPTION
           traitement des erreurs ]

        END;

- - fin du bloc pl/sql
/    - - exécution du bloc PL/SQL ; sans espace en début de ligne

- - autres commandes SQL pour sortir les résultats
```

Remarque : on peut créer une table temporaire en SQL, on l'utilise en PL/SQL pour y stocker des résultats, puis en SQL, à nouveau, on affiche les résultats et on détruit la table temporaire.
Exemple :

```
-- nombre d'artistes nés après une année donnée en paramètre
CREATE TABLE temp
  (nombre NUMBER(3));
DECLARE
  ANNEE NUMBER(4);
  Xnombre NUMBER(3);
BEGIN
  ANNEE:=&annee_naissance;
  SELECT COUNT(*) INTO Xnombre
   FROM A_artiste
   WHERE datns > ANNEE;
  INSERT INTO temp VALUES(Xnombre);
  COMMIT;
END;
/
SELECT * FROM temp;
```

5.2.2 Détails du script PL/SQL

PARTIE DECLARATION :

a) Variables

DECLARE nom_variable TYPE;

cas particuliers de types :

nom_variable nom_table.nom_colonne%TYPE ; (variable de même type que la colonne)

nom_variable nom_table%ROWTYPE ; (record avec les mêmes champs que la table)

variable initialisée

pays VARCHAR2(20):='FRANCE' ;

b) Constantes

DECLARE nom_constante CONSTANT TYPE;

pays CONSTANT VARCHAR2(20):='FRANCE' ;

La valeur de la constante ne peut pas être modifiée dans le bloc PlSql.

c) Curseurs (vus au chapître suivant)

```
DECLARE CURSOR nom_curseur IS
  SELECT nom_colonne(s)
   FROM nom_table(s)
   WHERE condition
   [ORDER BY nom_colonne(s)];
```

CORPS DU BLOC :

Bloc

Le corps d'un bloc PlSql est tout entier enfermé entre BEGIN et END;

S'il est vide, le mot BEGIN est omis mais END; demeure.

```
  [BEGIN ...] END;
```

Saisie de données au clavier avec &nom

Variables internes
   nom_variable TYPE;
   nom_variable TYPE := valeur_initiale;
Affectation
   :=
Insertion
   INSERT INTO nom_table VALUES (var_curseur.nom_colonnes(s))
Mise à jour d'une table :

```
DECLARE
BEGIN
  UPDATE f_agenda
    SET syn='francette',location = 'monet.org'
    WHERE syn = 'francette.monet';
 COMMIT;
END;
/
PROMPT 'Après mise à jour : ';
SELECT nom,syn||'@'||location
  FROM f_agenda ;

NOM
------------------------------
SYN||'@'||LOCATION
--------------------------------------------------
Francette Monet
fm@info.univ-angers.fr

Francette Monet
francette@monet.org
```

Ouverture, utilisation et fermeture d'un curseur
  OPEN nom_curseur... ;
  FETCH nom_curseur INTO nom_variable(s) ;
  UPDATE nom_table ... WHERE CURRENT OF CURSOR;
  CLOSE nom_curseur ;

5.3 - Structures algorithmiques Up

a) Boucle LOOP - END LOOP *"empaquetage des instructions"*

LOOP
  Bloc d'instructions avec

```
        EXIT WHEN condition;
      END LOOP;
```
Boucle FOR
```
      FOR condition -- l'indice de boucle n'est pas déclaré
        LOOP
          Bloc d'instructions;
        END LOOP;
```
Boucle WHILE
```
      WHILE condition
        LOOP
          Bloc d'instructions;
        END LOOP;
```
Alternatives/conditionnelles
```
      IF condition
        THEN Bloc_Instructions
        [ELSE
          Bloc_Instructions]
      END IF;
```
Alternatives emboîtées
```
      IF condition
        THEN Bloc_Instructions
        ELSIF condition
          THEN Bloc_Instructions
          ELSE Bloc_Instructions
      END IF;
```
Action nulle
```
        NULL ;
```
Séquence
```
      GOTO nom ;
      Bloc_Instructions ;
      ----------
      ----------
      <<nom>>
        Bloc_Instructions ;
```

5.4 - Saisie de données au clavier Up

&nom : saisie d'un mot au clavier ; un mot est un nombre ou une chaîne de caractères alphanumériques (éventuellement entre apostrophes)
Oracle substitura la suite des caractères saisis à &nom.
'&nom' ou '%&nom%' : permettront de ne saisir que les caractères indispensables.

Pour marquer le fin du paramètre de saisie, on peut mettre un point.

```
SELECT prénom, nom, telephone
FROM CLIENT
WHERE nom LIKE '%M&chaine.ET%';
```

5.5 - Affichages de résultats Up

On utilise le package (ensemble de procédures et fonctions) **DBMS_OUTPUT**.
Tout d'abord, avant le bloc PL/SQL, utiliser l'instruction :

set server output on

Puis pour chaque affichage :

dbms_output.put_line('texte'||X);

Nous remarquons la concaténation de chaînes de caractères avec : **||**

5.6 - Exemple

Exemple de bloc PL/SQL

Il s'agit d'automatiser une séquence d'Instructions.

A partir du code_client, déterminer sa dernière commande et calculer le montant de sa facture.

```
 SET SERVER OUTPUT ON

DECLARE
  xclient client%rowtype;
  xnum client.code_client%type;
  xnb NUMBER;
  xcom commande.no_commande%type;
  xmontant NUMBER(10,2);

BEGIN
  -- recherche du client
  xnum := '&code_du_client';
  SELECT * INTO xclient
    FROM client
    WHERE code_client = xnum;
  -- dernière commande
  SELECT COUNT(*) INTO xnb
    FROM commande
    WHERE code_client = xnum;
  IF xnb = 0
    THEN
```

```
    DBMS_OUTPUT.PUT_LINE(xclient.prenom||' '||xclient.nom||' n''a pas de
commande.');
  ELSE
    DBMS_OUTPUT.PUT_LINE(xclient.prenom||' '||xclient.nom);
    DBMS_OUTPUT.PUT_LINE(xclient.adresse||' '||xclient.code_postal||'
'||xclient.ville);
    -- dernière commande
    SELECT no_commande INTO xcom
      FROM commande
      WHERE code_client = xnum
      AND date_commande = (SELECT MAX(date_commande)
                           FROM commande
                           WHERE code_client = xnum);
    -- calcul du montant de la dernière facture
    SELECT SUM(prix*qte) INTO xmontant
      FROM article_commande ac, catalogue c
      WHERE ac.reference = c.reference
      AND ac.no_commande = xcom;
    DBMS_OUTPUT.PUT_LINE('Montant de la facture : '||xmontant||' F');
  ENDIF;
END;
/
```

5.7 - Exercices

FEUILLE D'EXERCICES SQL, SQL*PLUS N°05 : PLsql

1. BLOC PLsql, VALEURS SAISIES AU CLAVIER, ALTERNATIVE.
Considérons les artistes de deux pays choisis par
l'utilisateur.
Déterminons le nombre d'oeuvres et la moyenne des valeurs de ces oeuvres
pour chacun des deux pays.
Affichons les résultats : NB_OEUVRES VALEUR_MOYENNE précédés d'une phrase
dépendant du résultat, par exemple : les artistes du pays 1 vendent plus
cher que les artistes du pays 2, ou le contraire.

2. BLOC PLSQL - AFFICHAGE DES RESULTATS

Déterminer le nombre d'artistes et le nombre moyen d'oeuvres par artiste.

"En moyenne, les xxx artistes ont réalisé chacun yyy oeuvres."

3. BLOC PLSQL

Un nouvel artiste vient faire une exposition dans un musée répertorié.

- Quelles sont les tables concernées par cette action ?
- Créer et remplir ces tables à partir des tables fm.A_xxx.
- Ecrire un algorithme permettant d'insérer les nouvelles données dans les tables.
- Les nouvelles données sont saisies au clavier.
- Ces saisies et les mises à jour des tables sont faites dans un bloc PLSQL.
- Vérifier par des requêtes que les nouvelles données sont bien là.

3. BLOC PLSQL - CURSEUR - BOUCLE FOR...

Sélectionner les n premières oeuvres par ordre décroissant des
valeurs.

Remarque : les valeurs existent nécessairement.

code_oeuvre titre valeur

4. BLOC PLSQL - CURSEUR - BOUCLE LOOP ... END LOOP

Artistes ayant le plus grand nombre d'oeuvres répertoriées

nom date_naissance localite pays nb_oeuvres

5. Liste concaténée des artistes par musée, ayant exposé entre
deux dates données.

Chapitre 6 : PL/SQL - Curseurs

6.1 - Introduction

Une requête retourne un ensemble d'occurrences indissociables.
Il se peut que nous ayons besoin de traiter une partie seulement de
ces occurrences ou de ne retenir que les n premières occurrences trouvées.
Par exemple, trouver les trois meilleures commandes, modifier certaines des occurrences obtenues.

6.2 - Définition d'un curseur sans paramètre

Up

- Déclarations liées au curseur
- DECLARE
- -- déclaration et définition du curseur lui-même
- nom_curseur CURSOR IS
- SELECT ... ;
- -- déclarations des variables utiles pour l'utilisation du curseur,
- -- par exemple une variable LIGNE pour récupérer une occurrence du curseur.
- -- Il s'agit d'une variable de type enregistrement avec pour champs, les colonnes
- -- obtenues dans le curseur.
- LIGNE nom_curs%ROWTYPE ;
- Utilisation du curseur
- -- ouverture du curseur
- OPEN nom_curs ;
- *C'est à l'ouverture que la sélection est exécutée*
-
- -- Lecture de la ligne courante du curseur et stockage dans la variable ligne
- FETCH nom_curs INTO ligne ;
- -- Accès aux champs (ou colonnes) par ligne.nom_champ
- --

6.4 exercice

TPsql_5 : PLsql - Curseurs

1. BLOC PLsql, VALEURS SAISIES AU CLAVIER, ALTERNATIVE.
Considérons les artistes de deux pays choisis par
l'utilisateur.
Déterminons le nombre d'oeuvres et la moyenne des valeurs des
artistes pour chacun des deux pays.
Affichons par exemple que les artistes du pays 1 vendent plus
cher que les artistes du pays 2, ou le contraire.

2. BLOC PLSQL - AFFICHAGE DES RESULTATS
Déterminer le nombre d'artistes et le nombre moyen d'oeuvres par artiste.

"Les xxx artistes ont réalisé en moyenne yyy oeuvres."

3. BLOC PLSQL
Un nouvel artiste vient faire une exposition dans un musée répertorié.
- Quelles sont les tables concernées par cette action ?
- Créer et remplir ces tables à partir des tables fm.A_xxx.
- Ecrire un algorithme permettant d'insérer les nouvelles données dans les tables.
- Les nouvelles données sont saisies au clavier.
- Ces saisies et les mises à jour des tables sont faites dans un bloc PLSQL.
- Vérifier par des requêtes que les nouvelles données sont bien là.

3. BLOC PLSQL - CURSEUR - BOUCLE FOR...
Sélectionner les n premières oeuvres par ordre décroissant des
valeurs.
Remarque : les valeurs existent nécessairement.

code_oeuvre titre valeur

4. BLOC PLSQL - CURSEUR - BOUCLE LOOP ... END LOOP
Artistes ayant le plus grand nombre d'oeuvres répertoriées

nom date_naissance localite pays nb_oeuvres

5. Liste concaténée des artistes par musée, ayant exposé entre deux dates données.

Chapitre 8 : Oracle - Précompilateur C : proc

8.1 - Introduction

Oracle dispose d'un précompilateur C qui permet d'intégrer des commandes SQL dans un programme écrit en C.
Pendant la précompilation, Oracle génère des séquences de code qui insèrent les commandes SQL dans le programme C.
Le nom du fichier source a pour extension .pc.
La précompilation se fait avec la commande

```
proc iname=source.pc [options]
```

Le fichier peut être spécifié sans l'extension. On obtient un fichier de même nom et d'extension .c dans lequel les commandes SQL ont été traduites en instructions du langage C.

8.2 - Structure du programme

Up

```
/*  accès aux fichiers INCLUDE du langage C */
#include
/* partie déclaration */
EXEC SQL BEGIN DECLARE SECTION
- variables hôtes (utilisables dans les commandes SQL avec le préfixe : )
Exemple :
  VARCHAR xchaine(30);
  xchaine est une variable de type structuré :
    struct {unsigned short int len; /* longueur de la chaîne */
                    unsigned char arr[30];
                    }xchaine;
- les variables de connection
    VARCHAR utilisateur(20) = "nomlogin";
    VARCHAR mot_de_passe(20);
EXEC SQL END DECLARE SECTION;
/* inclusion du fichier de communication C - Oracle (bibliothèque proc) */
EXEC SQL INCLUDE SQLCA;
EXEC SQL INCLUDE ORACA;
/* corps du programme C avec la connection à Oracle, les instructions du programme
(instructions C et de commandes SQL (ou blocs PL/SQL)), la validation ou annulation des
actions SQL, la déconnection. */
```

EXEC SQL CONNECT :utilisateur [IDENTIFIED BY :mot_de_passe];

8.3 - Exemple Up

```
/* Define constants for VARCHAR lengths. */
#define     UNAME_LEN      20
#define     PWD_LEN        40

/* acces aux librairies standard d'entree-sorties */
/* equivalent du uses CRT du pascal */

#include

/* declaration des variables oracle/c */
EXEC SQL BEGIN DECLARE SECTION;
char libelle[40];
float valeur;
/* la variable uid sert a se connecter a l'ensemble ORACLE */
VARCHAR uid[UNAME_LEN];
VARCHAR pwd[PWD_LEN];
int counteur;
int compteur;
EXEC SQL INCLUDE ORACA;
EXEC SQL INCLUDE SQLCA;
EXEC SQL END DECLARE SECTION;

/* acces aux librairies Pro*c */
/* On definit en fait dans SQLCA toutes les fonctions et variables qui vont */
/* permettre aux compilateurs proC de traduire ce programme en C tout court */
/* afin qu'un simple cc puisse le compiler */
int compteur;
/* procedure principale et unique */
/* argc est le nombre de parametres passes au programme */
/* argv est le tableau d'emplacement de ces parametres */

void main(argc,argv)
int argc;
char *argv[];

{
```

```
/* connexion a Oracle */
   strncpy((char *) uid.arr, "monet", UNAME_LEN);
   uid.len = strlen((char *) uid.arr);
/* Copy the password.
*    strncpy((char *) pwd.arr, "*****", PWD_LEN);
*   pwd.len = strlen((char *) pwd.arr);
*   EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;
*/
/* Une petite interrogation sur la table A_artiste pour vérifier que l'on
 est bien connecté
*/
EXEC SQL SELECT count(*) into :counteur from A_artiste;
printf("le counteur donne %i\n", counteur);
for (compteur=1; compteur<5;compteur++)
{printf("coucou %i\n", compteur);
}

/* Validation des modifs */
/* EXEC SQL COMMIT;
*/

/* Deconnexion de la base */
/* EXEC SQL DISCONNECT;
*/

EXEC SQL COMMIT WORK RELEASE;

/* Attention aux problemes de Verrou si plusieurs programmes sont lances */
/* en meme temps */
}
```

8.4 - Compilation Up

Version utilisée :
Oracle8 Enterprise Edition Release 8.0.4.0.0 - Production
With the Objects option
PL/SQL Release 8.0.4.0.0 - Production

Pour compiler un fichier de nom nomfic.pc, taper :

make -f demo_proc.mk build OBJS="nomfic.o" EXE=nomfic

Attention ! Ne pas modifier le fichier demo_proc.mk

8.5 - Exercices

Chapître a : Oracle - Quelques commandes utiles - SQLLoader

l. Liste des tables auxquelles on peut accéder :

SELECT owner, table_name

FROM all_tables;

2. Liste des tables personnelles :

SELECT table_name

FROM user_tables;

ou

SELECT table_name

FROM tabs; -- tabs est un synonyme de user_tables.

3. Liste des synonymes de tables :

3.1. Rappelons la structure de la table all_catalog :

SQL>DESC all_catalog (entrée)

Name	Null?	Type
OWNER	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
TABLE_TYPE		VARCHAR2(11)

===> la colonne TABLE_TYPE permet de connaître le nom des synonymes de tables.

3.2 Sélection des occurrences contenant les noms des synonymes de tables :

SELECT *

FROM all_catalog

WHERE table_type == 'SYNONYM';

==> on peut voir les synonymes des tables 'Artistes'.

Par exemple, A_artiste est un synonyme de la table fm.A_artiste (accessible à tout public en lecture pour les requêtes).

Dans les requêtes, on peut remplacer fm.A_artiste. par son synonyme A_artiste

3.3 Requête sur la table fm.A_artiste à partir de son synonyme :

SELECT *

FROM A_artiste;
==> 52 occurrences

4. Création d'une table A_artiste dans son tablespace personnel :
4.1 Création de la structure :

```
Create Table A_artiste
  (cdart CHAR(2) PRIMARY KEY, /* code artiste*/
  nom VARCHAR (30),   /* Nom artiste */
  sexe CHAR (1),   /* sexe [l, 2] */
  datns VARCHAR ( 10 ) , /* Date ou année de naissance */
  localite VARCHAR(25), /* Localité de naissance */
  pays VARCHAR (15), /* Pays de naissance */
  datdc VARCHAR ( 10 ) , /* Date ou année de décès (sinon jour 1 et mois 1)*/
  cdmtr CHAR ( 2) /* Code maître */
  );
```

====> Cette table a le même nom que le synonyme. Elle masque l'accès à la table fm.A_artiste par son synonyme.

4.2 Contenu de la table A_artiste :

```
  SELECT *
    FROM A_artiste;
```

===> accès à la table de son propre tablespace, donc la table est vide.

```
  SELECT *
    FROM fm.A_artiste;
```

===> on retrouve les 52 occurrences de la table A_artiste existant dans le tablespace de fm.

5. Utilisation du module Sqlloader permettant de récupérer des données en provenance d'une autre Base de Données.

Exercice : Créer et remplir sous Oracle la table A_artiste que l'on avait auparavant sous Access.

5.1 La table Access a été exportée en fichier texte, avec " " comme délimiteurs de champs et des ; comme séparateurs de champs.

5.2 Création de la structure de la table A_artiste (cohérente bien sûr avec les données du fichier texte).

5.3 Adaptation d'un fichier X.ctl à la table que l'on veut récupérer. Voici par exemple le fichier art2.ctl :

```
load data
infile artiste, txt
into table A_artiste
fields terminated by ';' optionally enclosed by ""
(cdart, nom, sexe, datns, localite, pays, datdc, cdmtr)
```

5.4 Lancement **sous unix, en environnement Oracle, mais non sous sqlplus** de la commande :

```
sqlload 'login'/'mot_de_passe' X.ctl
```

==> des fichiers .log et .bad nous renseignent sur les erreurs rencontrées.

Travaux paratique

```
/*
  ===== CAVE D'UN VITICULTEUR ANGEVIN ======
*/

/*
  ===== LES VINS =====
 Les différents vins sont classés.
 - Il existe d'abord des classes de vins (ex. "vin de pays")
 - Chaque classe est divisée en appellations (ex. dans la classe "vin de pays",
 on trouve au moins les appellations "vin de table" et "Vin de Pays de Jardin
 de la France")
 - Chaque appellation est un ensemble de vins nommés (ex. l'appellation dite
 "Appellation d'Origine Controlée" contient les "Bonnezeaux","Anjou Villages",
 "Rosé de Loire", etc. pour ne citer que des vins élaborés chez ce Viticulteur
 Angevin)
 - Chaque vin nommé est subdivisé en "articles" tenant compte de l'année de
 fabrication du vin (le millésime) et du conditionnement (emballage),
*/

/*
  ===== CLIENTS ET TARIFS =====
 La clientèle est divisée en 4 catégories (ou types) : Particuliers, Foyers,
 Societes, Cafe-Hotel-Restaurant.
 Un tarif est adapté à chaque type de client.
*/

/*
  ===== ACHAT =====
 Chaque achat est enregistré sous forme de commande numérotée et datée
*/

/*
 ===== CREATION DES TABLES =====
*/

/*
```

```
 Table des appellations et classes de vins
*/
CREATE TABLE V_type_vins
  (appellation  varchar2(35) PRIMARY KEY, -- une appell. appartient à 1 classe
  classe        varchar2(12)              -- classe de vin = plusieurs appell.
  );

/*
 Table des noms de vins avec leur catégorie (blanc, rosé ou rouge)
*/
CREATE TABLE V_vin
  (nom          varchar2(25) PRIMARY KEY,                                    -- nom du vin (Anjou
Rouge)
  appellation   varchar2(35) REFERENCES V_type_vins (appellation) ,  -- référence à 1 appellat.
  categorie     varchar2(20)                                          -- BLANC, ROUGE ou ROSE
  );

/*
 Table des vins prets à la vente
*/
CREATE TABLE V_article
  (code_article varchar2(8) PRIMARY KEY,                                    -- identifie un vin
millésimé et conditionné
  nom           varchar2(25) REFERENCES V_vin(nom) ON DELETE CASCADE,      -- nom du vin
  millesime     number(2),                                                  -- millésime de ce vin
  emballage     varchar2(10)                                                -- BOUT, LITRES, FILLETTE,
MAGNUM
  );

/*
 Table des types de clientèle
*/
CREATE TABLE V_clientele
  (type_client number(1) PRIMARY KEY,     -- identifiant clientèle : de 1 à 4
  categ_client varchar2(22)               -- Particuliers, Foyers, Sociétés, Hôtels-Restaurants
  );

/*
 Table des références des clients, acheteurs de vin
*/
```

```
CREATE TABLE V_acheteur
 (no_acheteur number PRIMARY KEY,                                  -- identifiant acheteur de vin
 designation  varchar2(40),                                        -- nom, prénom ou label société, ...
 adresse      varchar2(50),                                        -- toute l'adresse dans une chaine
 type_client  number(1) REFERENCES V_clientele(type_client) -- identifie le type du client
 );

/*
 Table des tarifs dépendant des deux identifiants : article-vin et type-client
*/
CREATE TABLE V_tarif
 (code_article varchar2(8) REFERENCES V_article(code_article),          -- identifiant d'un vin
 type_client   number(1)REFERENCES V_clientele(type_client),            -- identifiant clientèle
 prixHT        number(7,2),                                              -- prix dépendant fonct. des
2 ident.
 CONSTRAINT c_tarif PRIMARY KEY (code_article, type_client)
 );

/*
 Table des achats enregistrés
*/
CREATE TABLE V_commande
 (no_commande  number PRIMARY KEY,                    -- identifiant achat d'un client
 date_commande date,                                  -- date achat
 no_acheteur   number REFERENCES V_acheteur(no_acheteur) -- identifiant acheteur
 );

/*
 Table des lignes de commandes pour un article-vin et un no-commande
*/
CREATE TABLE V_article_commande
 (code_article varchar2(8) REFERENCES V_article(code_article), -- identifiant d'un vin
 no_commande   number REFERENCES V_commande(no_commande),      -- identifiant achat
 quantite      number,                                         -- quantité de ce vin ds cet achat
 CONSTRAINT c_V_comm PRIMARY KEY (code_article, no_commande)
 );
```

```
/* insertions des donnees Vins */
/* insertion dans la table V_type_vins */

INSERT INTO V_type_vins VALUES ('Vin de Table','Vin de Pays');
INSERT INTO V_type_vins VALUES ('Vin de Pays de Jardin de la France','Vin de Pays');
INSERT INTO V_type_vins VALUES ('Appellation d''Origine Controlee','A.O.C.');

/* insertion dans la table V_vin */

INSERT INTO V_vin VALUES ('ANJOU BLANC 1/2 SEC','Vin de Table','BLANC');
INSERT INTO V_vin VALUES ('ANJOU BLANC CHENIN','Vin de Pays de Jardin de la
France','BLANC');
INSERT INTO V_vin VALUES ('ANJOU GAMAY','Appellation d''Origine Controlee','BLANC');
INSERT INTO V_vin VALUES ('ANJOU ROUGE','Appellation d''Origine Controlee','ROUGE');
INSERT INTO V_vin VALUES ('ANJOU VILLAGES','Appellation d''Origine Controlee','ROUGE');
INSERT INTO V_vin VALUES ('BONNEZEAUX','Appellation d''Origine Controlee','BLANC');
INSERT INTO V_vin VALUES ('CABERNET D''ANJOU','Appellation d''Origine Controlee','ROSE');
INSERT INTO V_vin VALUES ('COTEAUX DE L''AUBANCE','Appellation d''Origine
Controlee','ROUGE');
INSERT INTO V_vin VALUES ('ANJOU MOUSSEUX','Appellation d''Origine Controlee','BLANC');
INSERT INTO V_vin VALUES ('PINOT BLANC','Vin de Pays de Jardin de la France','BLANC');
INSERT INTO V_vin VALUES ('ROSE D''ANJOU','Appellation d''Origine Controlee','ROSE');
INSERT INTO V_vin VALUES ('ROSE DE LOIRE','Appellation d''Origine Controlee','ROSE');
INSERT INTO V_vin VALUES ('ROUGE VIN DE PAYS','Vin de Table','ROUGE');
INSERT INTO V_vin VALUES ('SAUVIGNON','Vin de Pays de Jardin de la France','BLANC');

/* insertion dans la table V_article */

INSERT INTO V_article VALUES ('ABD93L','ANJOU BLANC 1/2 SEC',93,'LITRES');
INSERT INTO V_article VALUES ('ABC93L','ANJOU BLANC CHENIN',93,'LITRES');
INSERT INTO V_article VALUES ('AG91B','ANJOU GAMAY',91,'BOUT');
INSERT INTO V_article VALUES ('AG93L','ANJOU GAMAY',93,'LITRES');
INSERT INTO V_article VALUES ('AG93B','ANJOU GAMAY',93,'BOUT');
INSERT INTO V_article VALUES ('AR91B','ANJOU ROUGE',91,'BOUT');
INSERT INTO V_article VALUES ('AR91F','ANJOU ROUGE',91,'FILLETTE');
INSERT INTO V_article VALUES ('AR92B','ANJOU ROUGE',92,'BOUT');
INSERT INTO V_article VALUES ('AR92L','ANJOU ROUGE',92,'LITRES');
INSERT INTO V_article VALUES ('AR92F','ANJOU ROUGE',92,'FILLETTE');
INSERT INTO V_article VALUES ('AR93B','ANJOU ROUGE',93,'BOUT');
INSERT INTO V_article VALUES ('AR93L','ANJOU ROUGE',93,'LITRES');
```

```
INSERT INTO V_article VALUES ('AV90','ANJOU VILLAGES',90,'BOUT');
INSERT INTO V_article VALUES ('AV91','ANJOU VILLAGES',91,'BOUT');
INSERT INTO V_article VALUES ('AV92MAG','ANJOU VILLAGES',92,'MAGNUM');
INSERT INTO V_article VALUES ('AV93','ANJOU VILLAGES',93,'BOUT');
INSERT INTO V_article VALUES ('BX89B','BONNEZEAUX',89,'BOUT');
INSERT INTO V_article VALUES ('BX90B','BONNEZEAUX',90,'BOUT');
INSERT INTO V_article VALUES ('BX91B','BONNEZEAUX',91,'BOUT');
INSERT INTO V_article VALUES ('BX92B','BONNEZEAUX',92,'BOUT');
INSERT INTO V_article VALUES ('BX93B','BONNEZEAUX',93,'BOUT');
INSERT INTO V_article VALUES ('BX93L','BONNEZEAUX',93,'LITRES');
INSERT INTO V_article VALUES ('CA92FIL', 'CABERNET D''ANJOU',92,'FILLETTE');
INSERT INTO V_article VALUES ('CA92B', 'CABERNET D''ANJOU',92,'BOUT');
INSERT INTO V_article VALUES ('CA92L', 'CABERNET D''ANJOU',92,'LITRES');
INSERT INTO V_article VALUES ('CA93L', 'CABERNET D''ANJOU',93,'LITRES');
INSERT INTO V_article VALUES ('CXA92B', 'COTEAUX DE L''AUBANCE',92,'BOUT');

INSERT INTO V_article VALUES ('CXA93B', 'COTEAUX DE L''AUBANCE',93,'BOUT');
INSERT INTO V_article VALUES ('CXA93L', 'COTEAUX DE L''AUBANCE',93,'LITRES');
INSERT INTO V_article VALUES ('CPH92B', 'ANJOU MOUSSEUX',92,'BOUT');
INSERT INTO V_article VALUES ('PIN92B', 'PINOT BLANC',92,'BOUT');
INSERT INTO V_article VALUES ('PIN92L', 'PINOT BLANC',92,'LITRES');
INSERT INTO V_article VALUES ('PIN93B', 'PINOT BLANC',93,'BOUT');
INSERT INTO V_article VALUES ('PIN93L', 'PINOT BLANC',93,'LITRES');
INSERT INTO V_article VALUES ('RA92L', 'ROSE D''ANJOU',92,'LITRES');
INSERT INTO V_article VALUES ('RA93L', 'ROSE D''ANJOU',93,'LITRES');
INSERT INTO V_article VALUES ('RL92B', 'ROSE DE LOIRE',92,'BOUT');
INSERT INTO V_article VALUES ('RL92L', 'ROSE DE LOIRE',92,'LITRES');
INSERT INTO V_article VALUES ('RL93L', 'ROSE DE LOIRE',93,'LITRES');
INSERT INTO V_article VALUES ('GR92L', 'ROUGE VIN DE PAYS',92,'LITRES');
INSERT INTO V_article VALUES ('GR93L', 'ROUGE VIN DE PAYS',93,'LITRES');
INSERT INTO V_article VALUES ('GR94L', 'ROUGE VIN DE PAYS',94,'LITRES');
INSERT INTO V_article VALUES ('SAU92B', 'SAUVIGNON',92,'BOUT');
INSERT INTO V_article VALUES ('SAU93L', 'SAUVIGNON',93,'LITRES');
INSERT INTO V_article VALUES ('SAU93B', 'SAUVIGNON',93,'BOUT');

/* table V_clientele = types de clients */

INSERT INTO V_clientele VALUES (1,'Particuliers');
INSERT INTO V_clientele VALUES (2,'Foyers');
INSERT INTO V_clientele VALUES (3,'Societes');
```

```
INSERT INTO V_clientele VALUES (4,'Cafe-Hotel-Restaurant');

/* table V_acheteur = les acheteurs */

INSERT INTO V_acheteur VALUES (1,'marchand catherine','26 rue hoche 45613 courville',1);
INSERT INTO V_acheteur VALUES (2,'fuclaux francois','6 impasse galiieni 63110 pinboche',1);
INSERT INTO V_acheteur VALUES (3,'clovis laurette','31 rue des ursulines 12560 leclos',1);
INSERT INTO V_acheteur VALUES (4,'tordose amelie','33 impasse des fleurs 44620 gorges',1);
INSERT INTO V_acheteur VALUES (5,'tarmiente victor','clos des vignes 69540 beaujolais',1);
INSERT INTO V_acheteur VALUES (6,'vanelant vanessa','42 lotissement des bieres 54770 basse-
yutz',1);
INSERT INTO V_acheteur VALUES (7,'ouagadougou mamadou','57 rue du niger 78950 bievres',1);

INSERT INTO V_acheteur VALUES (8,'la vie en rose','99 impasse geronte 75015 paris',2);
INSERT INTO V_acheteur VALUES (9,'les pins d''alep','62 rue du soudan 21540 gagneux',2);
INSERT INTO V_acheteur VALUES (10,'foyer maurice chevalier','26 avenue chapeau 92126
voigny',2);
INSERT INTO V_acheteur VALUES (11,'association des nonagenaires','13 rue souris 75114 paris',2);
INSERT INTO V_acheteur VALUES (12,'foyer dagobert','1 rue des ceinturons 73610 cotraz',2);
INSERT INTO V_acheteur VALUES (13,'foyer jules marey','23 rue des cellophanes 85220
bayeurley',2);
INSERT INTO V_acheteur VALUES (14,'societe manloxer','51 rue batracien 59450 batardeau',3);
INSERT INTO V_acheteur VALUES (15,'societe duchemole','64 avenue tempion 45780 tartifume',3);
INSERT INTO V_acheteur VALUES (16,'societe eaugrise','26 avenue clairefontaine 88200 saint-
die',3);
INSERT INTO V_acheteur VALUES (17,'societe tonusse','12 rue de padirac 65420 saint-flour',3);
INSERT INTO V_acheteur VALUES (18,'societe vanlaire','89 rue des eoliennes 69550
mangelevent',3);
INSERT INTO V_acheteur VALUES (19,'societe des roses','11 avenue inria 44850 cebonvivre',3);
INSERT INTO V_acheteur VALUES (20,'hotel bellevue','6 rue des fenetres 56254 zieutey',4);
INSERT INTO V_acheteur VALUES (21,'cafe hotel la bonne brise','29440 ouessant',4);
INSERT INTO V_acheteur VALUES (22,'hotel resto au bon coeur','33 rue michel colucci 78230
plaisir',4);
INSERT INTO V_acheteur VALUES (23,'cafe hotel resto a la vieille croute','76550 bonbeurre',4);
INSERT INTO V_acheteur VALUES (24,'pension familia','5 cours du cinematographe 54770
langenfeld',4);
INSERT INTO V_acheteur VALUES (25,'les mimosas','36 avenue du soleil 13200 brabassant',4);

/* table V_tarif (de */
```

```
INSERT INTO V_tarif VALUES ('ABC93L',1,9.69);
INSERT INTO V_tarif VALUES ('ABD93L',1,11.80);
INSERT INTO V_tarif VALUES ('AG91B',1,16.86);
INSERT INTO V_tarif VALUES ('AG93L',1,10.11);
INSERT INTO V_tarif VALUES ('AR91B',1,18.54);
INSERT INTO V_tarif VALUES ('AR92B',1,16.86);
INSERT INTO V_tarif VALUES ('AR92L',1,10.96);
INSERT INTO V_tarif VALUES ('AV90',1,20.23);
INSERT INTO V_tarif VALUES ('AV91',1,21.07);
INSERT INTO V_tarif VALUES ('AV92MAG',1,67.45);
INSERT INTO V_tarif VALUES ('BX89B',1,50.59);
INSERT INTO V_tarif VALUES ('BX90B',1,50.59);
INSERT INTO V_tarif VALUES ('BX91B',1,42.15);
INSERT INTO V_tarif VALUES ('BX92B',1,42.15);
INSERT INTO V_tarif VALUES ('BX93L',1,21.07);
INSERT INTO V_tarif VALUES ('CA92B',1,15.17);
INSERT INTO V_tarif VALUES ('CA93L',1,10.11);
INSERT INTO V_tarif VALUES ('CPH92B',1,26.98);
INSERT INTO V_tarif VALUES ('CXA92B',1,25.29);
INSERT INTO V_tarif VALUES ('CXA93L',1,15.17);
INSERT INTO V_tarif VALUES ('GR93L',1,6.32);
INSERT INTO V_tarif VALUES ('PIN92B',1,16.86);
INSERT INTO V_tarif VALUES ('PIN93L',1,9.69);
INSERT INTO V_tarif VALUES ('RA92L',1,7.58);
INSERT INTO V_tarif VALUES ('RL93L',1,9.27);
INSERT INTO V_tarif VALUES ('SAU92B',1,18.54);
INSERT INTO V_tarif VALUES ('SAU93B',1,18.54);
INSERT INTO V_tarif VALUES ('SAU93L',1,11.80);

INSERT INTO V_tarif VALUES ('AG93B',2,12.94);
INSERT INTO V_tarif VALUES ('AG93L',2,8.43);
INSERT INTO V_tarif VALUES ('AR92B',2,13.78);
INSERT INTO V_tarif VALUES ('AR92F',2,7.58);
INSERT INTO V_tarif VALUES ('AR93B',2,13.78);
INSERT INTO V_tarif VALUES ('AR93L',2,9.27);
INSERT INTO V_tarif VALUES ('AV90',2,16.86);
INSERT INTO V_tarif VALUES ('AV92MAG',2,42.15);
INSERT INTO V_tarif VALUES ('AV93',2,16.86);
INSERT INTO V_tarif VALUES ('BX91B',2,31.19);
INSERT INTO V_tarif VALUES ('BX92B',2,31.19);
```

```
INSERT INTO V_tarif VALUES ('CA92B',2,10.96);
INSERT INTO V_tarif VALUES ('CPH92B',2,26.98);
INSERT INTO V_tarif VALUES ('CXA93B',2,18.54);
INSERT INTO V_tarif VALUES ('CXA93L',2,15.17);
INSERT INTO V_tarif VALUES ('GR94L',2,6.32);
INSERT INTO V_tarif VALUES ('PIN93B',2,10.11);
INSERT INTO V_tarif VALUES ('RA93L',2,7.16);
INSERT INTO V_tarif VALUES ('RL92B',2,10.11);
INSERT INTO V_tarif VALUES ('SAU93B',2,16.86);

INSERT INTO V_tarif VALUES ('ABC93L',3,7.58);
INSERT INTO V_tarif VALUES ('AR91F',3,7.58);
INSERT INTO V_tarif VALUES ('AR92B',3,13.49);
INSERT INTO V_tarif VALUES ('AR92L',3,9.27);
INSERT INTO V_tarif VALUES ('CA92FIL',3,6.74);
INSERT INTO V_tarif VALUES ('CA92B',3,11.8);
INSERT INTO V_tarif VALUES ('CA92L',3,8.43);
INSERT INTO V_tarif VALUES ('CXA92B',3,21.92);
INSERT INTO V_tarif VALUES ('CPH92B',3,26.98);
INSERT INTO V_tarif VALUES ('PIN92L',3,7.58);
INSERT INTO V_tarif VALUES ('RA92L',3,7.16);
INSERT INTO V_tarif VALUES ('GR93L',3,6.32);

INSERT INTO V_tarif VALUES ('AR91B',4,14.33);
INSERT INTO V_tarif VALUES ('AR92B',4,14.33);
INSERT INTO V_tarif VALUES ('AR92L',4,9.27);
INSERT INTO V_tarif VALUES ('AV90',4,16.86);
INSERT INTO V_tarif VALUES ('AV91',4,18.54);
INSERT INTO V_tarif VALUES ('BX91B',4,33.72);
INSERT INTO V_tarif VALUES ('BX92B',4,33.72);
INSERT INTO V_tarif VALUES ('CA92B',4,12.64);
INSERT INTO V_tarif VALUES ('CPH92B',4,26.98);
INSERT INTO V_tarif VALUES ('PIN92B',4,14.33);
INSERT INTO V_tarif VALUES ('RL92B',4,11.8);
INSERT INTO V_tarif VALUES ('RL92L',4,9.27);
INSERT INTO V_tarif VALUES ('GR93L',4,5.90);


/* table V_commande */
```

```
INSERT INTO V_commande VALUES (1,'21-JUN-96',3);
INSERT INTO V_commande VALUES (2,'24-JUN-96',2);
INSERT INTO V_commande VALUES (3,'30-JUN-96',1);
INSERT INTO V_commande VALUES (4,'10-JUL-96',10);
INSERT INTO V_commande VALUES (5,'12-JUL-96',15);
INSERT INTO V_commande VALUES (6,'15-JUL-96',10);
INSERT INTO V_commande VALUES (7,'18-JUL-96',4);
INSERT INTO V_commande VALUES (8,'24-JUL-96',17);
INSERT INTO V_commande VALUES (9,'28-JUL-96',21);
INSERT INTO V_commande VALUES (10,'28-JUL-96',16);
INSERT INTO V_commande VALUES (11,'02-AUG-96',11);
INSERT INTO V_commande VALUES (12,'04-AUG-96',20);
INSERT INTO V_commande VALUES (13,'06-AUG-96',5);
INSERT INTO V_commande VALUES (14,'12-AUG-96',6);
INSERT INTO V_commande VALUES (15,'15-AUG-96',7);
INSERT INTO V_commande VALUES (16,'15-AUG-96',18);
INSERT INTO V_commande VALUES (17,'20-AUG-96',22);
INSERT INTO V_commande VALUES (18,'30-AUG-96',13);
INSERT INTO V_commande VALUES (19,'03-SEP-96',8);
INSERT INTO V_commande VALUES (20,'10-SEP-96',9);
INSERT INTO V_commande VALUES (21,'15-SEP-96',12);
INSERT INTO V_commande VALUES (22,'20-SEP-96',13);
INSERT INTO V_commande VALUES (23,'10-OCT-96',19);
INSERT INTO V_commande VALUES (24,'20-OCT-96',20);
INSERT INTO V_commande VALUES (25,'28-OCT-96',24);
INSERT INTO V_commande VALUES (26,'29-NOV-96',14);
INSERT INTO V_commande VALUES (27,'14-DEC-96',23);
INSERT INTO V_commande VALUES (28,'30-DEC-96',25);
INSERT INTO V_commande VALUES (29,'15-JAN-97',16);
INSERT INTO V_commande VALUES (30,'21-JAN-97',25);
INSERT INTO V_commande VALUES (31,'27-FEB-97',22);


/* table V_article_commande */

INSERT INTO V_article_commande VALUES ('AR92B',1,6);
INSERT INTO V_article_commande VALUES ('SAU92B',1,12);
INSERT INTO V_article_commande VALUES ('AR91B',2,3);
INSERT INTO V_article_commande VALUES ('BX92B',2,3);
INSERT INTO V_article_commande VALUES ('SAU93B',2,3);
```

```
INSERT INTO V_article_commande VALUES ('BX89B',3,6);
INSERT INTO V_article_commande VALUES ('CA93L',3,32);
INSERT INTO V_article_commande VALUES ('AR92F',4,12);
INSERT INTO V_article_commande VALUES ('CXA93L',4,32);
INSERT INTO V_article_commande VALUES ('SAU93B',4,6);
INSERT INTO V_article_commande VALUES ('ABC93L',5,32);
INSERT INTO V_article_commande VALUES ('CA92B',5,6);
INSERT INTO V_article_commande VALUES ('PIN92L',5,10);
INSERT INTO V_article_commande VALUES ('CPH92B',6,10);
INSERT INTO V_article_commande VALUES ('BX89B',7,6);
INSERT INTO V_article_commande VALUES ('BX90B',7,6);
INSERT INTO V_article_commande VALUES ('AR91F',8,12);
INSERT INTO V_article_commande VALUES ('AR91B',9,12);
INSERT INTO V_article_commande VALUES ('AV90',9,24);
INSERT INTO V_article_commande VALUES ('BX92B',9,12);
INSERT INTO V_article_commande VALUES ('GR93L',9,64);
INSERT INTO V_article_commande VALUES ('AR92B',10,6);
INSERT INTO V_article_commande VALUES ('PIN92L',10,32);

INSERT INTO V_article_commande VALUES ('AR93B',11,12);
INSERT INTO V_article_commande VALUES ('AV90',11,12);
INSERT INTO V_article_commande VALUES ('GR94L',11,64);
INSERT INTO V_article_commande VALUES ('CA92B',12,6);
INSERT INTO V_article_commande VALUES ('BX92B',12,24);
INSERT INTO V_article_commande VALUES ('CPH92B',12,12);
INSERT INTO V_article_commande VALUES ('RL92L',12,64);
INSERT INTO V_article_commande VALUES ('AR91B',13,12);
INSERT INTO V_article_commande VALUES ('PIN92B',14,6);
INSERT INTO V_article_commande VALUES ('SAU92B',14,6);

INSERT INTO V_article_commande VALUES ('BX93L',15,10);
INSERT INTO V_article_commande VALUES ('CA92L',16,32);
INSERT INTO V_article_commande VALUES ('AR92L',16,32);
INSERT INTO V_article_commande VALUES ('AR92L',17,64);
INSERT INTO V_article_commande VALUES ('GR93L',17,128);
INSERT INTO V_article_commande VALUES ('AV93',18,6);
INSERT INTO V_article_commande VALUES ('CA92B',18,12);
INSERT INTO V_article_commande VALUES ('AR93L',19,32);
INSERT INTO V_article_commande VALUES ('CXA93L',19,32);
INSERT INTO V_article_commande VALUES ('AV92MAG',20,16);
```

```
INSERT INTO V_article_commande VALUES ('GR94L',21,32);
INSERT INTO V_article_commande VALUES ('RA93L',21,32);
INSERT INTO V_article_commande VALUES ('SAU93B',21,6);
INSERT INTO V_article_commande VALUES ('AR93B',22,6);
INSERT INTO V_article_commande VALUES ('GR94L',22,32);
INSERT INTO V_article_commande VALUES ('AR92B',23,12);
INSERT INTO V_article_commande VALUES ('CA92B',23,12);
INSERT INTO V_article_commande VALUES ('PIN92L',23,10);
INSERT INTO V_article_commande VALUES ('BX91B',24,24);
INSERT INTO V_article_commande VALUES ('BX92B',24,24);
INSERT INTO V_article_commande VALUES ('PIN92B',24,12);
INSERT INTO V_article_commande VALUES ('AR92L',25,32);
INSERT INTO V_article_commande VALUES ('RL92L',25,32);
INSERT INTO V_article_commande VALUES ('CA92L',26,48);
INSERT INTO V_article_commande VALUES ('AR92L',27,64);
INSERT INTO V_article_commande VALUES ('GR93L',27,32);

INSERT INTO V_article_commande VALUES ('BX91B',28,12);
INSERT INTO V_article_commande VALUES ('BX92B',28,12);
INSERT INTO V_article_commande VALUES ('PIN92B',28,6);
INSERT INTO V_article_commande VALUES ('PIN92L',29,32);
INSERT INTO V_article_commande VALUES ('RA92L',29,32);
INSERT INTO V_article_commande VALUES ('CA92B',30,6);
INSERT INTO V_article_commande VALUES ('AV91',30,12);
INSERT INTO V_article_commande VALUES ('AR92L',31,32);
INSERT INTO V_article_commande VALUES ('RL92L',31,32);
```

```
/* suppression des tables vins : l'ordre de suppression dépend des dépendances fonctionnelles */

DROP TABLE V_article_commande;

DROP TABLE V_commande;

DROP TABLE  V_tarif;

DROP TABLE V_acheteur;

DROP TABLE V_clientele;

DROP TABLE V_article;

DROP TABLE V_vin;

DROP TABLE V_type_vins;
```

/ * TP2 - Vins - Requêtes de base, jointures, ordre */

SUJET

Une cave viticole en Anjou

REQUETES

1. Afficher les noms des vins blancs (de categorie BLANC).

NOM

-- rangement des résultats au niveau de l'affichage

2. Afficher la liste des appellations et des noms des vins ordonnés selon l'appellation.

APPELLATION NOM

-- Jointure et rangement des résultats

3. Afficher la liste des noms de vins, du conditionnement, de la catégorie ordonnés selon la catégorie et dans chaque catégorie selon le conditionnement.

NOM EMBALLAGE CATEGORIE

4. Afficher la liste des clients avec le numéro et la date de leurs commandes. Ordonner les résultats selon la désignation du client et la date de commande.

DESIGNATION NO_COMMANDE DATE_COMMANDE

-- colonne calculée, alias de colonne, (arrondi ; voir chapître sur les fonctions

5. Afficher le nom du vin, le millésime, le prix TTC pour les vins en bouteille vendus aux particuliers.
Ordonner les résultats selon le nom, le millésime (ordre décroissant).

NOM MILLESIME PRIX

-- Utilisation des opérateurs de l'algèbre relationnelle

6. Déterminer les vins (nom, millésime, conditionnement) vendus aux Particuliers et non aux Sociétés.
Ordonner selon le conditionnement, le nom.

EMBALLAGE NOM MILLESIME

/ * TP3 - Vins - Fonctions, Agrégation de données, Mise en page des résultats */

SUJET

Une cave viticole en Anjou

REQUETES

1. Afficher le montant de la vente TTC de chaque vin (defini par son nom).

2. Afficher le montant de la vente TTC par catégorie de vin.

3. Afficher la liste des désignations des acheteurs 'Particuliers' avec le montant global de leurs commandes.

4. Afficher le nombre de litres de vin vendus, pour chaque vin (défini par son nom, son millésime), sachant qu'une bouteille contient 0,75 litre, une fillette 0,375 litre et un magnum 1,5 litre.
Le nom du vin ne sera pas répété.

5. Liste des différents noms de vins classés par catégorie et rangés dans l'ordre alphabétique. A l'affichage, la catégorie n'est pas répétée, une ligne blanche sépare les catégories.

6. Calculer le montant TTC des ventes par année civile.
Affichage par ordre chronologique des années.

-- Utiliser Having
7. Afficher les prix moyens HT des différentes vins définis par leur nom et conditionnés en bouteilles sachant que l'on s'intéresse à une seule catégorie d'acheteurs.

/ * TP4 - Vins -Sous-Requêtes /

REQUETES

1. Existe-t-il des vins (nom, millésime, conditionnement) vendus exclusivement
à une catégorie de clients ? Si oui, en établir la liste par type de clientèle.

2. Calculer le montant TTC des ventes, par année civile.

3. Afficher le nom, le millésime, le nombre de bouteilles de l'article
(conditionné en bouteille) le plus commandé.
Remarque : en réponse, il peut y avoir plusieurs articles.

--Vues
4. Nous voulons faire parvenir une publicite aux cafes-hotels-restaurant en leur
envoyant le tarif HT qui leur convient avec le code_article, le nom, le millesime,
l'emballage, le prix HT.Creer une vue contenant tous ces renseignements.

5. Nous voulons établir la facture correspondant a un numéro de commande donné.
Créer une vue pour chaque point suivant :
- les coordonnées de l'acheteur
- l'ensemble des lignes de la facture avec : nom du vin, millesime, quantite, emballage,
montant TTC.
Déterminer le montant total de la facture.

6. Afficher, pour les acheteurs particuliers et pour le dernier millésime pour lequel
des particuliers ont fait des achats, la liste des noms de vin, leur conditionnement
(emballage), leur prix TTC (la TVA est a 19.6%).
Ranger les résultats selon la categorie et le nom du vin.
Faire une rupture avec saut de ligne pour chaque catégorie.

-- dates, sous-requête, fonctions
7. Liste des clients avec la date de leur dernière commande.

-- date, saisie au clavier, sous-requête
8. On veut envoyer un encart publicitaire aux clients qui n'ont rien commandé depuis
une date donnée.
Afficher la liste de ces clients (avec leurs coordonnées).

-- fonctions, vues

8. On veut faire profiter d'une promotion les clients qui ont passé commande pour plus de X Francs depuis une date donnée.
La date et la valeur X sont saisies au clavier.
Afficher la liste de ces clients (avec leurs coordonnées).