Réponses aux ateliers SQL

Atelier 1 Présentation de l'environnement

Questions

1-1. Une table peut-elle avoir plusieurs clés primaires ?

Réponse : Une table ne peut comporter qu'une seule clé primaire, même lorsque celle-ci est constituée d'une combinaison de plusieurs colonnes.

1-2. Une table peut-elle avoir une contrainte unique si elle possède déjà une clé primaire ?

Réponse : Il est possible de spécifier une contrainte unique pour une colonne de clé non primaire afin de garantir que toutes les valeurs de cette colonne seront uniques.

1-3. Une table qui possède une clé étrangère est-elle une table enfant ou une table parent ?

Réponse : Elle est définie dans des tables enfant et assure qu'un enregistrement parent a été créé avant un enregistrement enfant et que l'enregistrement enfant sera supprimé avant l'enregistrement parent.

1-4. Que signifie LMD ?

Réponse : Le Langage de Manipulation de Données et de modules, ou LMD (en anglais DML), pour déclarer les procédures d'exploitation et les appels à utiliser dans les programmes.

1-5. Que signifie LDD ?

Réponse : Le Langage de Définition de Données ou LDD (en anglais DDL), à utiliser pour déclarer les structures logiques de données et leurs contraintes d'intégrité.

1-6. Quels sont les types d'instructions qui ne peuvent être exécutés en PL/SQL ?

Réponse : Le langage PL/SQL ne comporte pas d'instructions du Langage de Définition de Données « ALTER », « CREATE », « RENAME » et d'instructions de contrôle comme « GRANT » et « REVOKE ».

1-7. Quels sont les avantages du langage PL/SQL par rapport au SQL ?

Réponse : Le langage PL/SQL combine la puissance de manipulation des données du SQL avec la puissance de traitement d'un langage procédural. Il offre de nombreux avantages : support de la programmation orientée objet, très bonnes performances, portabilité, facilité de programmation, parfaite intégration à Oracle et à Java.

1-8. Pour configurer le client, lequel de ces fichiers utilisez-vous?

A. init.ora

B. sqlnet.ora

C. listener.ora

D. tnsnames.ora

Réponse : D

1-9. Quel est le répertoire où se trouvent les fichiers de configuration ?

A. %ORACLE_HOME%\admin\network

B. %ORACLE_HOME%\network\admin

C. %ORACLE_HOME%\net90\admin

Réponse : B

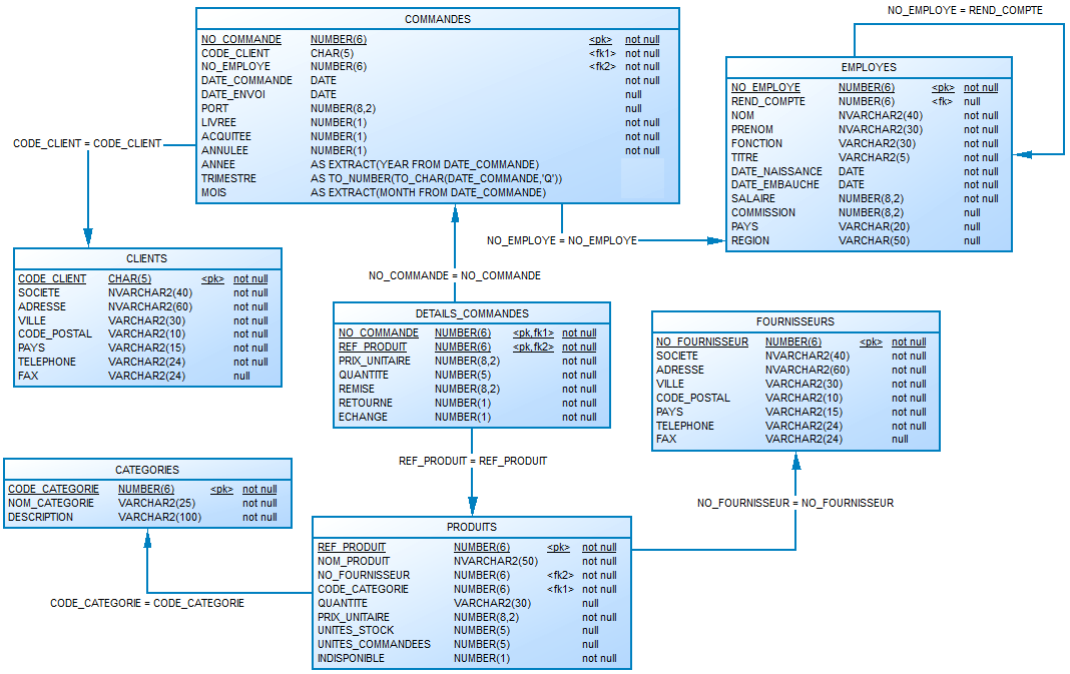
Exercice n° 1 Installation

Installez Oracle XE sur votre machine en tenant compte de votre système d'exploitation.

Exercice n° 2 Les tables utilisées pour les ateliers

Sachant que le symbole <pk> signifie clé primaire et <fk> la clé étrangère.

Quelles sont les tables en relation parent enfant ?



Réponse :

Parent	**Enfant**
CATEGORIES	PRODUITS
FOURNISSEURS	PRODUITS
PRODUITS	DETAILS_COMMANDES
COMMANDES	DETAILS_COMMANDES
CLIENTS	COMMANDES
EMPLOYES	COMMANDES
EMPLOYES	EMPLOYES

Atelier 1.2 Les outils SQL*Plus

Questions

1. Quel est l'outil que vous retrouvez sur chaque serveur de base de données installée ?

   A. SQL*Plus.

   B. iSQL*Plus.

   C. SQL*Plus Worksheet

   D. Oracle Enterprise Manager.

   Réponse : A

2. SQL*Plus est-il un langage ou un environnement ?

   Réponse : Un environnement

3. Pour utiliser iSQL*Plus sur une machine distante, avez-vous besoin d'installer le client Oracle ?

   Réponse : Non

4. Quelle est la commande qui vous permet de vous connecter ?

   Réponse : CONNECT

5. Dans la syntaxe de démarrage de SQL*Plus, pouvez-vous lancer l'exécution d'un script ?

   Réponse : OUI

6. Quelle est la commande qui vous permet de stocker dans un fichier tout ce qui est affiché à l'écran ?

   Réponse : SPOOL

7. Dans l'environnement SQL*Plus, peut-on exécuter des commandes du système d'exploitation ?

   Réponse : OUI

8. Citez trois types de paramètres de mise en forme des résultats des requêtes.

   Réponse : LINSEIZE, PAGESIZE, FEEDBACK

9. Quelle est la commande qui vous permet de décrire la structure d'une vue ?

   Réponse : DESC

Exercice n° 1 Préparer le poste de développement

Installez le schéma des exemples pour les ateliers en respectant la démarche suivante :

```
C:\>dir Oracle11gSQL_PLSQL.zip
 Le volume dans le lecteur C n'a pas de nom.
 Le numéro de série du volume est BC79-154D

 Répertoire de C:\
```

```
12/08/2011  20:34         6 787 143 Oracle11gSQL_PLSQL.zip

C:\>unzip Oracle11gSQL_PLSQL.zip
Archive:  Oracle11gSQL_PLSQL.zip
   creating: Oracle11gSQL_PLSQL/
  inflating: Oracle11gSQL_PLSQL/DeleteEnvStagiaireXE.sql
  inflating: Oracle11gSQL_PLSQL/InitEnvEtoileXE.sql
  inflating: Oracle11gSQL_PLSQL/InitEnvStagiaireXE.sql
   creating: Oracle11gSQL_PLSQL/stagiaire/
  inflating: Oracle11gSQL_PLSQL/stagiaire/CATEGORIES.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/CLIENTS.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/COMMANDES.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/COMMANDES_2009.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/DETAILS_COMMANDES.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/DETAILS_COMMANDES_2009.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/DIM_TEMPS.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/EMPLOYES.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/FOURNISSEURS.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/PRODUITS.DAT
  inflating: Oracle11gSQL_PLSQL/stagiaire/STATISTIQUES.DAT

C:\>cd Oracle11gSQL_PLSQL

C:\Oracle11gSQL_PLSQL>dir

 Répertoire de C:\Oracle11gSQL_PLSQL

12/08/2011  21:52    <REP>          .
12/08/2011  21:52    <REP>          ..
12/08/2011  21:52               550 DeleteEnvStagiaireXE.sql
12/08/2011  21:51             1 725 InitEnvEtoileXE.sql
12/08/2011  20:33            30 681 InitEnvStagiaireXE.sql
12/08/2011  20:33    <REP>          stagiaire

C:\Oracle11gSQL_PLSQL>slqplus /nolog @InitEnvStagiaireXE.sql
```

Téléchargez et Installez l'outil SQL Developer.

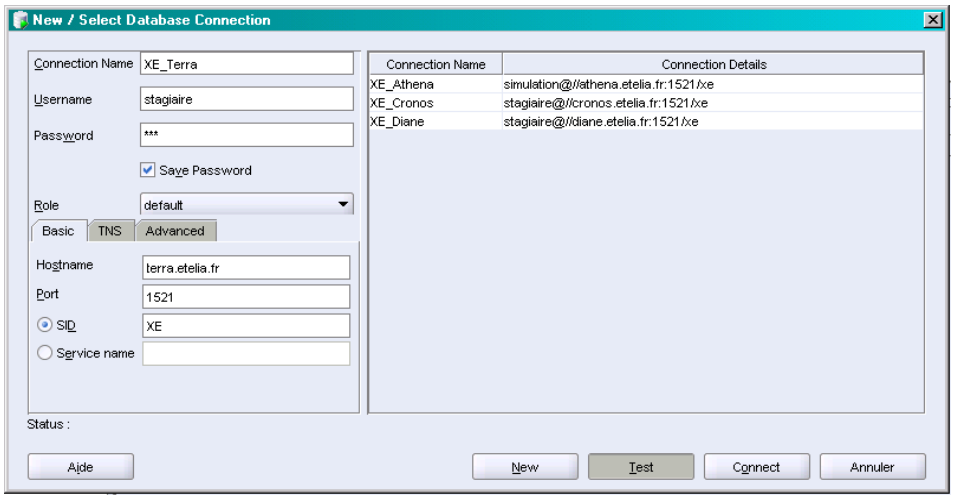
Exercice n° 2 Connexion

Démarrez SQL*Plus, en ligne de commande, avec le nom d'utilisateur du schéma exemples « **STAGIAIRE** » et son mot de passe « **PWD** ».

```
C:\>sqlplus stagiaire/pwd
...
C:\>sqlplus stagiaire/pwd@//diane.etelia.fr:1521/XE
...

C:\>sqlplus /nolog
SQL> CONNECT STAGIAIRE/PWD
ou
SQL> stagiaire/pwd@//diane.etelia.fr:1521/XE
SQL> show user
USER est "STAGIAIRE"
```

Démarrez SQL Developer et paramétrez la connexion à la base de données.



Exercice n° 3 Environnement SQL*Plus

En utilisant SQL*Plus en ligne de commande, redirigez les sorties vers un fichier et exécutez les commandes suivantes :

- Décrivez la table **« COMMANDES »** ;

```
C:>sqlplus stagiaire/pwd
SQL> SPOOL C:\Exercice2.lst
SQL> DESC COMMANDES
 Nom                           NULL ?   Type
 ----------------------------- -------- ------------------
 NO_COMMANDE                   NOT NULL NUMBER(6)
 CODE_CLIENT                   NOT NULL CHAR(5)
 NO_EMPLOYE                    NOT NULL NUMBER(6)
 DATE_COMMANDE                 NOT NULL DATE
 DATE_ENVOI                             DATE
 PORT                                   NUMBER(8,2)
```

- Déconnectez-vous de la base de données sans sortir du SQL*Plus ;

```
SQL> DISC
Déconnecté de Oracle Database 11g Express Edition Release 11.2.0.1.0
- Production
```

- Décrivez de nouveau la table **« COMMANDES »**. Que remarquez-vous ?

```
SQL> DESC COMMANDES
SP2-0640: Non connecté
SP2-0641: "DESCRIBE" nécessite une connexion au serveur
```

- Connectez vous ;

```
SQL> CONNECT STAGIAIRE/PWD
ou
SQL> stagiaire/pwd@//diane.etelia.fr:1521/XE
```

- Affichez l'utilisateur courant ;

```
SQL> show user
USER est "STAGIAIRE"
```

- Arrêtez la redirection des sorties vers le fichier ;

```
SQL> SPOOL OFF
```

- Sans quitter l'environnement, listez le fichier que vous venez de créer.

```
SQL> HOST TYPE C:\Exercice2.lst
```

Exercice n°4 Générer des scripts SQL

Connectez-vous à SQL*Plus, redirigez les sorties vers le fichier **« DESC_ALL.SQL »** et exécutez les commandes suivantes :

- Interrogez la vue catalogue à l'aide de la syntaxe suivante :

```
SET PAGESIZE 0
SET ECHO OFF
SET FEEDBACK OFF
SELECT 'DESC '||TABLE_NAME FROM CAT
WHERE TABLE_TYPE = 'TABLE' ;
```

```
SQL> SET PAGESIZE 0
SQL> SET ECHO OFF
SQL> SET FEEDBACK OFF
SQL> SPOOL C:\DESC_ALL.SQL
SQL> SELECT 'DESC '||TABLE_NAME FROM CAT
  2  WHERE TABLE_TYPE = 'TABLE' ;
```

- Maintenant vous pouvez arrêter la redirection des sorties vers le fichier et exécuter le script ainsi conçu.

```
SQL> SPOOL OFF
SQL> @C:\DESC_ALL.SQL
```

Atelier 2.1 Interrogation des données

Questions

1. Lesquelles de ces syntaxes sont correctes ?

   A. `SELECT * FROM CAT;`

   B. `SeLeCt * From Employes;`

   C. `SELECT ALL FONCTION FROM EMPLOYES;`

   D. `SELECT DISTINCT * FROM CATEGORIES;`

   E. `select nom client from employes;`

   F. `select 'L''employe', NOM  from employes;`

   G. `select nom, salaire*1.2 from employes;`

   H. `select salaire*1,2*commission from employes;`

   I. `select NOM "Employé" from employes;`

   J. `SELECT 'bonjour !!' "C'est un :" from employes;`

   Réponse : Toutes

2. Quel est le symbole de fin de bloc SQL ?

   Réponse : « ; »

3. Quel est le symbole de délimitation des chaînes de caractères ?

   Réponse : « ' »

4. Quel est la seule utilisation du caractère « " » ?

   Réponse : Uniquement pour délimiter un alias.

Exercice n° 1 Projection totale

Écrivez les requêtes vous permettant d'afficher :

– Les employés de la société.

```
SQL> SELECT * FROM EMPLOYES;
```

– Les catégories de produits.

```
SQL> SELECT * FROM PRODUITS;
```

– Les enregistrements de n'importe quelle table saisie au démarrage de la requête.

```
SQL> SELECT * FROM &NOM_TABLE;
Entrez une valeur pour nom_table : FOURNISSEURS
```

Exercice n° 2 Projection

Écrivez les requêtes vous permettant d'afficher :

– Le nom, le prénom et la date de naissance de tous les employés de la société.

```
SQL> SELECT NOM, PRENOM, DATE_NAISSANCE FROM EMPLOYES;
```

– Le nom de la société, la ville et le pays de tous les fournisseurs.

```
SQL> SELECT SOCIETE, VILLE, PAYS FROM FOURNISSEURS;
```

– La fonction de tous les employés.

```
SQL> SELECT FONCTION FROM EMPLOYES;
```

– Toutes les fonctions des employés de l'entreprise, chaque fonction doit être affichée une seule fois.

```
SQL> SELECT DISTINCT FONCTION FROM EMPLOYES;
```

– La liste des localités dans lesquelles la société a au moins un client.

```
SQL> SELECT DISTINCT VILLE FROM CLIENTS;
```

Atelier 2.2 Interrogation des données

Questions

1. Laquelle de ces syntaxes est incorrecte ?

   A. `SELECT NOM||' '||PRENOM "Employé" FROM EMPLOYES;`

   B. `SELECT NOM||" "||PRENOM 'Employé' FROM EMPLOYES;`

   C. `SELECT NOM||' '||SALAIRE "Employé" FROM EMPLOYES;`

   D. `SELECT SALAIRE*1.10 FROM EMPLOYES;`

   E. `SELECT PORT||DATE_ENVOI FROM COMMANDES;`

   F. `SELECT SALAIRE*1,10*COMMISSION FROM EMPLOYES;`

   G. `SELECT DATE_EMBAUCHE-DATE_NAISSANCE FROM EMPLOYES;`

   H. `SELECT DATE_EMBAUCHE - 10 FROM EMPLOYES;`

   I. `SELECT DATE_EMBAUCHE + 1/24 FROM EMPLOYES;`

   J. `SELECT 'bonjour !!' "C'est un :" from employes;`

   Réponse : B

2. Quelle est la fonction qui traite les valeurs « **`NULL`** » d'une expression ?

   Réponse : `NVL`

3. Sachant que la colonne `COMMISSION` peut ne pas être renseignée, comportée des valeurs « **`NULL`** ». Quelle est la syntaxe qui affiche une valeur pour chaque employé ?

   A. `SELECT NOM,SALAIRE*COMMISSION FROM EMPLOYES;`

   B. `SELECT NOM,SALAIRE*(COMMISSION,0)FROM EMPLOYES;`

   C. `SELECT NOM,SALAIRE*NVL(COMMISSION,0)FROM EMPLOYES;`

   D. `SELECT NOM,SALAIRE*NVL(COMMISSION)FROM EMPLOYES;`

   Réponse : C

Exercice n° 1 Concaténation

Respectant les formats des modèles suivants, écrivez les requêtes vous permettant d'afficher :

– Le nom de l'employé et ses revenues annuels : commission + salaire * 12.

```
Employé              a un  gain annuel sur 12 mois
-------------------- ----- ----------- -------------
Fuller               gagne      120000 par an.
Buchanan             gagne       96000 par an.
...
```

```
SQL> SELECT NOM "Employé",
  2         'gagne' "a un",
```

```
  3         nvl(COMMISSION,0) + SALAIRE*12 "gain annuel",
  4         'par an.     ' "sur 12 mois"
  5  FROM EMPLOYES;
```

– Le nom et le prénom de l'employé et sa fonction.

```
Employé
------------------------------------------------------------
Callahan Laura est Assistante commerciale de cette société.
Buchanan Steven est Chef des ventes de cette société.
...
```

```
SQL> SELECT NOM||' '||PRENOM||' est '||FONCTION||
  2                 ' de cette société.' "Employé"
  3  FROM EMPLOYES;
```

Exercice n° 2 Opérateurs

Créez les requêtes vous permettant d'afficher :

– Les produits commercialisés, la valeur du stock par produit et la valeur des produits commandés. Dans la table PRODUITS, vous trouvez les champs UNITES_STOCK et UNITES_COMMANDEES que vous multipliez par le PRIX_UNITAIRE pour retrouver les valeurs des deux stocks.

```
SQL> SELECT NOM_PRODUIT,
  2         PRIX_UNITAIRE*UNITES_STOCK "Stock",
  3         UNITES_COMMANDEES*PRIX_UNITAIRE "Commandes"
  4  FROM PRODUITS;
```

– Le nom, le prénom, l'âge et l'ancienneté des employés, dans la société.

```
SQL> SELECT NOM,
  2         PRENOM,
  3         SYSDATE - DATE_NAISSANCE "Age",
  4         SYSDATE - DATE_EMBAUCHE  "Ancienneté"
  5  FROM EMPLOYES;
```

– Le numéro de la commande, le temps écoulé entre la commande et la livraison de celle-ci ainsi que les frais de port.

```
SQL> SELECT NO_COMMANDE,
  2         nvl( DATE_ENVOI - DATE_COMMANDE, -1 )
  3                              "Durée de livraison",
  4         PORT "Frais de port"
  5  FROM COMMANDES;
```

Atelier 2.3 Interrogation des données

Questions

1. Quelles sont les requêtes qui affichent les employés triés par ordre croisant de leur fonction et dans le cadre d'une même fonction, d'abord les employés qui gagnent le plus ?

   A. `SELECT FONCTION, SALAIRE, NOM, PRENOM FROM EMPLOYES ORDER BY NOM, PRENOM;`

   B. `SELECT FONCTION, SALAIRE, NOM, PRENOM FROM EMPLOYES ORDER BY FONCTION, SALAIRE DESC;`

   C. `SELECT FONCTION, SALAIRE, NOM, PRENOM FROM EMPLOYES ORDER BY FONCTION, SALAIRE;`

   D. `SELECT FONCTION, SALAIRE, NOM, PRENOM FROM EMPLOYES ORDER BY 1, 2 DESC;`

   Réponse : B, D

2. Quelle est la valeur de tri par défaut ?

   A. `ASC`

   B. `DESC`

   Réponse : A

3. les valeurs « **`NULL`** » sont-elles affichées d'abord ou en dernier ?

   Réponse : Oracle traite les valeurs « **`NULL`** » comme si elles étaient des valeurs infinies, les lignes correspondantes sont affichées en dernier.

Exercice n° 1 Les ordres de tri

Écrivez les requêtes permettant d'afficher :

– Les employés par ordre alphabétique.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
  2  FROM EMPLOYES
  3  ORDER BY NOM;
```

– Les employés depuis le plus récemment embauché jusqu'au plus ancien.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
  2  FROM EMPLOYES
  3  ORDER BY DATE_EMBAUCHE DESC;
```

– Les fournisseurs dans l'ordre alphabétique de leur pays et ville de résidence.

```
SQL> SELECT SOCIETE, VILLE, PAYS
  2  FROM FOURNISSEURS
  3  ORDER BY PAYS, VILLE;
```

– Les employés par ordre alphabétique de leur fonction et du plus grand salaire au plus petit.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
  2  FROM EMPLOYES
```

```
  3  ORDER BY FONCTION, SALAIRE DESC;
```

– Les employés dans l'ordre de leur commission.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE, COMMISSION
  2  FROM EMPLOYES
  3  ORDER BY COMMISSION NULLS FIRST;
```

Exercice n° 2 Les pseudocolonnes

Affichez l'utilisateur connecté et la date du jour comme le modèle suivant.

```
Bonjour Utilisateur                Aujourd'hui                Date
------- ------------------------- ------------------------- --------
Bonjour STAGIAIRE                  Aujourd'hui nous sommes : 17/04/06
```

```
SQL> SELECT 'Bonjour' "Bonjour", user "Utilisateur",
  2    'Aujourd''hui nous sommes :' "Aujourd'hui",
  3    SYSDATE "Date"
  4  FROM DUAL;
```

Atelier 3.1 Les opérateurs logiques

Questions

1. Quelles sont les opérateurs logiques qui peuvent être utilisés avec les chaînes de caractères ?

   A. `=`

   B. `>`

   C. `<`

   D. `!=, ^=, < >`

   E. `LIKE`

   F. `REGEXP_LIKE`

   Réponse : Tous

2. Pour les mêmes choix que la question précédente, quelles sont les opérateurs logiques qui peuvent être utilisés avec les dates ?

   Réponse : A, B, C, D

3. Quelle est l'opérateur de répétition qui permet une mise en correspondance avec n'importe quel caractère unique appartenant à la liste ?

   A. `^`

   B. `$`

   C. `*`

   D. `|`

   E. `[]`

   F. `()`

   G. `{}`

   H. `\`

   I. `?`

   Réponse : E

4. Pour les mêmes choix que la question précédente, quel est l'opérateur de répétition qui permet une mise en correspondance avec la chaîne correspondante ?

   Réponse : F

Exercice n° 1 La restriction

Écrivez les requêtes permettant d'afficher :

– Le nom de la société et de la localité des clients qui habitent à Toulouse.

```
SQL> SELECT SOCIETE, VILLE FROM CLIENTS
  2  WHERE VILLE = 'Toulouse';
```

– Le nom, le prénom et la fonction des employés qui ne sont pas des représentants.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
  2  WHERE FONCTION <> 'Représentant(e)';
```

– Le nom du produit, la catégorie et le fournisseur des produits qui ne sont pas disponibles, le champ `INDISPONIBLE` est égal à 1.

```
SQL> SELECT NOM_PRODUIT,CODE_CATEGORIE, NO_FOURNISSEUR
  2  FROM PRODUITS
  3  WHERE INDISPONIBLE = 1;
```

– Le nom, prénom et fonction des employés qui ont un salaire inférieur à 3500.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
  2  WHERE SALAIRE < 3500;
```

– Le nom, prénom et fonction des employés dirigés par l'employé numéro 86.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
  2  WHERE REND_COMPTE = 86;
```

– Le nom, prénom et fonction des employés recrutés après 01/01/2003.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
  2  WHERE  DATE_EMBAUCHE > '01/01/2003';
```

Exercice n° 2 Le traitement des chaînes de caractères

Écrivez les requêtes permettant d'afficher :

– Les produits et leur quantité conditionnée en bouteilles d'un litre.

```
SQL> SELECT NOM_PRODUIT, QUANTITE FROM PRODUITS
  2  WHERE QUANTITE LIKE '%bouteille%1%litre%';
```

– Le nom de la société cliente, la localité et le code postal des fournisseurs à condition que leur code postal soit composé uniquement des valeurs numériques.

```
SQL> SELECT SOCIETE, VILLE, CODE_POSTAL FROM FOURNISSEURS
  2  WHERE REGEXP_LIKE(CODE_POSTAL,'^[[:digit:]]+$');
```

– Les produits et leur quantité à condition que leur emballage soit de type cartons, boîtes ou unités et conditionnée par paquets de 24 ou 32.

```
SQL> SELECT NOM_PRODUIT, QUANTITE FROM PRODUITS
  2  WHERE REGEXP_LIKE ( QUANTITE,
  3                      '^(24|32).(carton|pièces|bouteilles)');
```

– Le nom de la société, le pays et le numéro de téléphone à condition que leur numéro de téléphone soit formaté de la sorte : **`'99.99.99.99.99'`**.

```
SQL> SELECT SOCIETE, PAYS, TELEPHONE  FROM CLIENTS
  2  WHERE REGEXP_LIKE ( TELEPHONE,
  3                     '^[[0-9]{2}(.[0-9]{2}){4}$');
```

– Le nom de la société cliente, le pays et le numéro de téléphone à condition que leur numéro de téléphone commence soit par, **`'(604)'`**, **`'(91)'`** ou **`'(5)'`**.

```
SQL> SELECT SOCIETE, PAYS, TELEPHONE  FROM CLIENTS
  2  WHERE REGEXP_LIKE ( TELEPHONE, '^\((604|91|5)\)');
```

– Les produits et leur quantité à condition que leur emballage est type bouteille ou pots et leur poids soit mentionné en onces ou litres.

```
SQL> SELECT NOM_PRODUIT, QUANTITE FROM PRODUITS
  2  WHERE REGEXP_LIKE ( QUANTITE,
  3                      '(bouteille|pot).*(litre|once)');
```

Exercice n° 3 Le traitement de valeurs NULL

Écrivez les requêtes permettant d'afficher :

– Le nom de la société, la ville et le pays des clients qui n'ont pas de numéro de fax renseigné.

```
SQL> SELECT SOCIETE, VILLE, PAYS
  2  FROM CLIENTS
  3  WHERE FAX IS NULL;
```

– Le nom, prénom et la fonction des employés qui ne sont pas commissionnés.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
  2  WHERE COMMISSION IS NULL;
```

– Le nom, prénom et la fonction des employés qui n'ont pas de supérieur hiérarchique.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
  2  WHERE REND_COMPTE IS NULL;
```

– Le nom de la société, la ville et le pays des fournisseurs qui ont un numéro de fax renseigné.

```
SQL> SELECT SOCIETE, VILLE, PAYS
  2  FROM FOURNISSEURS
  3  WHERE FAX IS NOT NULL;
```

Atelier 3.2 Les opérateurs logiques

Questions

1. Quelles sont les conditions qui permettront d'afficher les produits livrés par le fournisseur numéro 1 et numéro 2 ?

   A. `NO_FOURNISSEUR BETWEEN 1 AND 2`

   B. `NO_FOURNISSEUR = 1 AND NO_FOURNISSEUR = 2`

   C. `NO_FOURNISSEUR = 1 OR NO_FOURNISSEUR = 2`

   D. `NO_FOURNISSEUR IN (1,2)`

   Réponse : A, C, D

2. Quelle est la condition qui permettra d'afficher les produits de la catégorie 1 ou de la catégorie 2 livrés par le fournisseur numéro 1 ?

   A.
   ```
   NO_FOURNISSEUR = 1 AND
   CODE_CATEGORIE = 1 OR CODE_CATEGORIE = 2
   ```

   B.
   ```
   ( NO_FOURNISSEUR = 1 AND
     CODE_CATEGORIE = 1 ) OR CODE_CATEGORIE = 2
   ```

   C.
   ```
   NO_FOURNISSEUR = 1 AND
   ( CODE_CATEGORIE = 1 OR CODE_CATEGORIE = 2 )
   ```

   D.
   ```
   NO_FOURNISSEUR = 1 OR
   CODE_CATEGORIE = 1 OR CODE_CATEGORIE = 2
   ```

   Réponse : C

3. Quelle est la condition qui permettra d'afficher les commandes qui n'ont pas encore été livrées et leur frais de port qui ont été renseigné ?

   A. `DATE_ENVOI IS NULL OR PORT IS NULL`

   B. `DATE_ENVOI IS NULL OR PORT IS NOT NULL`

   C. `DATE_ENVOI IS NULL AND PORT IS NOT NULL`

   Réponse : C

Exercice n° 1 L'opérateur BETWEEN

Écrivez les requêtes permettant d'afficher :

– Le nom, prénom, fonction et salaire des employés qui ont un salaire compris entre 3500 et 6000.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
  2  FROM EMPLOYES
  3  WHERE  SALAIRE  BETWEEN 3500 AND 6000;
```

– Le numéro de commande, code client et la date de commande pour les commandes passées entre le `'01/01/2011'` et `'03/01/2011'`.

```
SQL> SELECT NO_COMMANDE, CODE_CLIENT, DATE_COMMANDE
  2  FROM COMMANDES
  3  WHERE DATE_COMMANDE BETWEEN '01/01/2011' AND '03/01/2011';
```

Exercice n° 2 La comparaison avec des listes

Écrivez les requêtes permettant d'afficher :

– Le nom de la société, l'adresse, le téléphone et la ville des clients qui habitent à Toulouse, à Strasbourg, à Nantes ou à Marseille.

```
SQL> SELECT SOCIETE, ADRESSE, TELEPHONE, VILLE
  2  FROM CLIENTS
  3  WHERE VILLE IN ('Toulouse','Strasbourg','Nantes','Marseille');
```

– Le nom du produit, le fournisseur, la catégorie et les quantités en stock pour les produits qui sont d'une des catégories 1, 3, 5 et 7.

```
SQL> SELECT NOM_PRODUIT, NO_FOURNISSEUR,
  2         CODE_CATEGORIE, UNITES_STOCK
  3  FROM PRODUITS
  4  WHERE CODE_CATEGORIE IN (1,3,5,7);
```

– Le numéro de commande, code client et la date de commande pour les commandes passées dans une des dates : `'18/02/2011'`, `'20/02/2011'` ou `'25/02/2011'`.

```
SQL> SELECT NO_COMMANDE,CODE_CLIENT,DATE_COMMANDE
  2  FROM COMMANDES
  3  WHERE DATE_COMMANDE IN
  4        ('18/02/2011','20/02/2011','25/02/2011');
```

Exercice n° 3 L'assemblage des expressions

Écrivez les requêtes permettant d'afficher :

– Le nom, prénom, fonction et le salaire des représentants qui sont en activité depuis `'10/10/2002'`.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
  2  FROM EMPLOYES
  3  WHERE DATE_EMBAUCHE > '10/10/2002' AND
  4        FONCTION LIKE 'Repr%';
```

– Le nom, prénom, fonction et le salaire des employés qui sont âgés de plus de 45 ans ou qui ont une ancienneté de plus de 10 ans.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
  2  FROM EMPLOYES
  3  WHERE (SYSDATE - DATE_NAISSANCE)/365 > 45 or
  4        (SYSDATE - DATE_EMBAUCHE)/365 > 10;
```

– Le nom du produit, le fournisseur, la catégorie et les quantités des produits qui ont le numéro fournisseur entre 1 et 3 ou un code catégorie entre 1 et 3 et pour lesquelles les quantités sont données en boîtes ou en cartons.

```
SQL> SELECT NOM_PRODUIT, NO_FOURNISSEUR,
  2         CODE_CATEGORIE, UNITES_STOCK
  3  FROM PRODUITS
  4  WHERE NO_FOURNISSEUR BETWEEN 1 AND 3 AND
  5        CODE_CATEGORIE BETWEEN 1 AND 3 AND
  6        REGEXP_LIKE ( QUANTITE,'(boîtes|cartons)');
```

– Les produits et leur quantité à condition que leur emballage ne soit pas d'un de ces types : cartons, boîtes ou unités et qu'il ne soit pas conditionné par paquets de 24 ou 32. Il ne faut pas afficher les produits de catégorie 1, 4 et 8.

```
SQL> SELECT NOM_PRODUIT, QUANTITE FROM PRODUITS
```

```
2  WHERE NOT( REGEXP_LIKE ( QUANTITE,
3                  '^(24|32)[ ](carton|pièces|bouteilles)') AND
4              CODE_CATEGORIE IN (1,4,8));
```

Atelier 4.1 Les chaînes de caractères

Questions

1. Quelle fonction permet de convertir en majuscule la première lettre de chaque mot de la chaîne ?

   Réponse : `INITCAP`

2. Quelles sont les syntaxes incorrectes ?

   A. `SELECT CONCAT(NOM,' ',PRENOM) FROM EMPLOYES ;`

   B. `SELECT CONCAT(NOM,PRENOM) FROM EMPLOYES ;`

   C. `SELECT NOM||' '||PRENOM FROM EMPLOYES ;`

   D. `SELECT CONCAT(NOM,' ')||PRENOM FROM EMPLOYES ;`

   Réponse : A

3. Quelles sont les requêtes qui permettent d'afficher le résultat suivant ?

```
Format
---------------
xxxxxxxxxFULLER
xxxxxxxBUCHANAN
xxxxxxxxPEACOCK
xxxxxxLEVERLING
xxxxxxxxDAVOLIO
xxxxxxDODSWORTH
xxxxxxxxxxxKING
xxxxxxxxxSUYAMA
xxxxxxxCALLAHAN
```

   A. `SELECT LPAD(UPPER(NOM), 15,'x') FROM EMPLOYES ;`

   B. `SELECT UPPER(LPAD(NOM, 15,'x')) FROM EMPLOYES ;`

   C. `SELECT LPAD('xxxxxxx'||UPPER(NOM)) FROM EMPLOYES ;`

   D. `SELECT 'xxxxxxxxxxx'||UPPER(NOM) FROM EMPLOYES`

   Réponse : A, B

4. Quel est le résultat de la requête suivante ?

```
SQL> SELECT DISTINCT SUBSTR( QUANTITE, INSTR(QUANTITE,' '),
  2                                    INSTR(QUANTITE,' ',1,2) -
  3                                    INSTR(QUANTITE,' ') )
  4  FROM PRODUITS ;
```

   A.

```
Expression
------------------------------
bouteille (500 ml)
bouteille (750 cc)
bouteilles (0,5 litre)
bouteilles (1 litre)
bouteilles (12 onces)
```

```
bouteilles (250 ml)
...
```

B.

```
Expression
------------------------------
1 b
1 c
10 b
10 c
10 s
10 v
...
```

C.

```
Expression
------------------------------
bouteille
bouteilles
boîtes
canettes
carton
pièces
plaquettes
pots
sacs
unités
verres
```

Réponse : C

5. Quelle fonction vous permet d'effacer plusieurs caractères parasites positionnés au début de la chaîne ?

   Réponse : `LTRIM`

6. Quelles fonctions vous permettent d'effacer plusieurs caractères parasites positionnés n'importe où dans la chaîne ?

   Réponse : `REGEXP_REPLACE`, `TRANSLATE`

7. Quelles fonctions vous permettent de remplacer une chaîne de caractères par un caractère ?

   Réponse : `REGEXP_REPLACE`, `REPLACE`

Exercice n° 1 Le formatage des chaînes

Écrivez les requêtes permettant d'afficher :

– Le nom et le prénom en majuscule concaténées avec un espace au milieu. Il faut prendre soin de ne pas dépasser une longueur maximum de 14 caractères.

```
SQL> SELECT LPAD( UPPER( NOM ||' '|| PRENOM ), 14 ) "Employé"
  2  FROM EMPLOYES;
```

Exercice n° 2 La manipulation des chaînes

Écrivez les requêtes permettant d'afficher :

– La liste des produits, type d'emballage (boîte, boîtes, pots, cartons, ...) et quantité du type d'emballage ('36 boîtes', '12 pots (12 onces)', ...) triés par ordre alphabétique du type d'emballage. Le résultat de la requête doit être comme dans l'exemple suivant :

```
NOM_PRODUIT                      Emballage                Quantité
-------------------------------- ------------------------ --------
Konbu                            boîtes                          1
Chai                             boîtes                         10
Zaanse koeken                    boîtes                         10
Teatime Chocolate Biscuits       boîtes                         10
Ipoh Coffee                      boîtes                         16
Filo Mix                         boîtes                         16
Alice Mutton                     boîtes                         20
Boston Crab Meat                 boîtes                         24
Pâté chinois                     boîtes                         24
Pavlova                          boîtes                         32
...
```

```
SQL> SELECT QUANTITE,
  2         SUBSTR( QUANTITE, INSTR(QUANTITE,' '),
  3                           INSTR(QUANTITE,' ',1,2) -
  4                           INSTR(QUANTITE,' ') ),
  5         SUBSTR( QUANTITE, 1,
  6                           INSTR(QUANTITE,' ') )
  7  FROM PRODUITS;
```

– Les employés et leur âge comme dans l'exemple suivant :

```
Employé                        Âge
------------------------------ ---
FULLER Andrew                  50
BUCHANAN Steven                47
CALLAHAN Laura                 44
PEACOCK Margaret               43
KING Robert                    41
LEVERLING Janet                38
SUYAMA Michael                 38
DAVOLIO Nancy                  33
DODSWORTH Anne                 32
```

```
SQL> SELECT UPPER(NOM)||' '||PRENOM "Employé",
  2         SUBSTR((SYSDATE - DATE_NAISSANCE)/365,1,2) "Âge"
  3  FROM EMPLOYES;
```

ou

```
SQL> SELECT UPPER(NOM)||' '||PRENOM "Employé",
  2         REGEXP_REPLACE( (SYSDATE - DATE_NAISSANCE)/365,',(.*)')
  3  FROM EMPLOYES;
```

– La société et le numéro de téléphone des fournisseurs comme une liste des valeurs numériques.

```
SQL> SELECT SOCIETE,
  2         REGEXP_REPLACE(TELEPHONE,'[^0-9]') "Téléphone"
  3  FROM FOURNISSEURS;
```

Atelier 4.2 Les fonctions numériques

Questions

1. Quelles sont les syntaxes incorrectes ?

   A. `SELECT 2.5f, 2.5D FROM DUAL;`

   B. `SELECT 2.5/0, 2.5f, 2.5D FROM DUAL;`

   C. `SELECT 2.5, 2.5f/0, 2.5D/0 FROM DUAL;`

   D. `SELECT 2.5*2.5/0f, 2.5D/0 FROM DUAL;`

   E. `SELECT NVL(2.5*2.5/0f), 2.5D/0 FROM DUAL;`

   Réponse : B, E

2. Quelle fonction vous permet de remplacer une expression si elle n'est pas une valeur numérique ou si elle n'a pas de valeur, par une valeur significative ?

   Réponse : `NANVL`

3. Quelles sont les opérateurs logiques pour travailler avec les expressions à virgule flottante qui n'ont pas des valeurs exploitables ?

   Réponse : `IS NAN`, `IS INFINITE`

4. Quelle est la valeur renvoyée par l'expression suivante :

```
ROUND(TRUNC(MOD(1600,10),-1),2)
```

   A. `NULL`

   B. `1`

   C. `0`

   D. `0.00`

   E. Une erreur

   Réponse : C

Exercice n° 1 Le formatage des chaînes

Écrivez les requêtes permettant d'afficher :

– Les employés et leur salaire journalier (salaire / 20) arrondi à l'entier inférieur.

```
SQL> SELECT NOM, PRENOM, FLOOR(SALAIRE/20)
  2  FROM EMPLOYES;
```

– Les employés et leur salaire journalier (salaire / 20) arrondi à l'entier supérieur.

```
SQL> SELECT NOM, PRENOM, CEIL(SALAIRE/20)
  2  FROM EMPLOYES;
```

– Les produits commercialisés, la valeur du stock, les unités en stock fois le prix unitaire, arrondie à la centaine près.

```
SQL> SELECT NOM_PRODUIT, ROUND( PRIX_UNITAIRE*UNITES_STOCK, -2)
  2   FROM PRODUITS;
```

– Les produits commercialisés, la valeur du stock, les unités en stock fois le prix unitaire, arrondie à la dizaine inférieure.

```
SQL> SELECT NOM_PRODUIT, TRUNC( PRIX_UNITAIRE*UNITES_STOCK, -1)
  2   FROM PRODUITS;
```

– Les employés et leur revenu annuel (salaire*12 + commission) arrondi à la centaine près.

```
SQL> SELECT NOM, PRENOM, ROUND( SALAIRE*12 + NVL(COMMISSION,0),-2)
  2  FROM EMPLOYES;
```

Atelier 4.3 Le traitement des dates

Questions

1. Quelle est la date renvoyée par la fonction suivante : `ADD_MONTHS('30/10/2006',4)` ?

   A. `NULL`

   B. `29/02/2007;`

   C. `28/02/2007;`

   D. `01/03/2007;`

   Réponse : C

2. Quelle est la fonction qui renvoie la date et l'heure relative à la plage horaire de la session. ?

   A. `CURRENT_DATE`

   B. `CURRENT_TIMESTAMP`

   C. `DBTIMEZONE`

   D. `SESSIONTIMEZONE`

   E. `LOCALTIMESTAMP`

   F. `SYSTIMESTAMP`

   G. `SYS_EXTRACT_UTC`

   H. `TZ_OFFSET`

   Réponse : B

3. Pour les mêmes choix que la question précédente, quelle est la pseudocolonne qui indique le fuseau horaire de la session ?

   Réponse : `D`

4. Pour les mêmes choix que la question précédente, quelle est la fonction permettant de renvoyer l'écart de fuseau horaire entre la zone passée en paramètre et UTC c'est-à-dire à l'heure de Greenwich. ?

   Réponse : `H`

5. Quelles sont les opérations invalides ?

   A. `DATE + INTERVAL`

   B. `DATE - INTERVAL`

   C. `TIMESTAMP + NUMBER`

   D. `TIMESTAMP - NUMBER`

   E. `TIMESTAMP + TIMESTAMP`

   F. `TIMESTAMP - TIMESTAMP`

   G. `TIMESTAMP + INTERVAL`

   H. `TIMESTAMP – INTERVAL`

I. INTERVAL * NUMBER

J. INTERVAL / NUMBER

K. TIMESTAMP * INTERVAL

L. TIMESTAMP / INTERVAL

Réponse : C, E, K, L

Exercice n° 1 Les zones horaires

Écrivez les requêtes permettant d'afficher :

– Le fuseau horaire du serveur et le fuseau horaire de la session. Changez votre fuseau horaire de la session pour 'Europe/Athens' et affichez de nouveau le fuseau horaire du serveur et le fuseau horaire de la session.

```
SQL> SELECT DBTIMEZONE, SESSIONTIMEZONE FROM DUAL;
SQL> ALTER SESSION SET TIME_ZONE = 'Europe/Athens';
SQL> SELECT DBTIMEZONE, SESSIONTIMEZONE FROM DUAL;
```

– La date et l'heure actuelle en prenant en compte la zone horaire configurée sur la session, le fuseau horaire du serveur et le fuseau horaire de la session.

```
SQL> SELECT CURRENT_DATE, DBTIMEZONE, SESSIONTIMEZONE FROM DUAL;
```

– La date et l'heure actuelle en prenant en compte la zone horaire configurée sur le serveur, le fuseau horaire du serveur et le fuseau horaire de la session.

```
SQL> SELECT SYSTIMESTAMP, DBTIMEZONE, SESSIONTIMEZONE FROM DUAL;
```

Exercice n° 2 La manipulation des dates

Écrivez les requêtes permettant d'afficher :

– La date du prochain dimanche (à ce jour).

```
SQL> SELECT NEXT_DAY(SYSDATE,'dimanche') FROM DUAL;
```

– Les dates du premier et du dernier jour du mois en cours.

```
SQL> SELECT TRUNC(SYSDATE,'MM'), LAST_DAY(SYSDATE) FROM DUAL;
```

– La date du premier jour du trimestre (format 'Q').

```
SQL> SELECT TRUNC(SYSDATE,'Q') FROM DUAL;
```

– Le nom, la date de fin de période d'essai (3 mois) et leur ancienneté à ce jour exprimé en mois pour tous les employés.

```
SQL> SELECT NOM,
  2         ADD_MONTHS( DATE_EMBAUCHE,3),
  3         TRUNC( MONTHS_BETWEEN( SYSDATE, DATE_EMBAUCHE))
  4  FROM EMPLOYES;
```

– Le nom et le jour de leur première paie (dernier jour du mois de leur embauche).

```
SQL> SELECT NOM,
  2         LAST_DAY(DATE_EMBAUCHE)
  3  FROM EMPLOYES;
```

Atelier 5.1 Les conversions SQL

Questions

1. Quelles sont les syntaxes incorrectes ?

   A. `TO_CHAR (2000, '#,###.##U')`

   B. `TO_CHAR (2000, '0,000.00U')`

   C. `TO_CHAR (2000, '9,999.00U')`

   D. `TO_CHAR (2000, '9,999.99U')`

   E. `TO_CHAR (2000, '2,000.00U')`

   F. `TO_CHAR (2000, 'N,NNN.NNU')`

   G. `TO_CHAR (2000, '#,###.##U')`

   H. `TO_CHAR (2000, '0,000.00U')`

   I. `TO_CHAR (2000, '9G999D00L')`

   J. `TO_CHAR (2000, '9G999D99L')`

   Réponse : A, E, F, G

2. Quelles sont les syntaxes correctes ?

   A. `TO_CHAR(SYSDATE,'DD, DY Month, YYY')`

   B. `TO_CHAR(SYSDATE,'FMDay, DD Month, YYYY')`

   C. `TO_CHAR(SYSDATE,'Day, DD Month, YYYY')`

   D. `TO_DATE(SYSDATE,'Day, DD Month, YYYY')`

   E. `TO_CHAR(SYSDATE,'FMDY, DDD Month, YYYY')`

   F. `TO_DATE(SYSDATE,'DY, DDD Month, YYYY')`

   Réponse : A, B, C, E

3. Pour les mêmes choix que la question précédente, quelle est la fonction permettant de renvoyer la date formatée de la sorte : `'Vendredi, 14 Juillet, 2006'` ?

   Réponse : `B`

Exercice n° 1 Les conversions

Écrivez les requêtes permettant d'afficher :

– La date du jour formatée de la sorte :

```
Nous sommes le :
------------------------
Jeudi 11 Août 2011
```

```
SQL> SELECT TO_CHAR(SYSDATE,'FMDay DD Month YYYY')
  2  "Nous sommes le :"  FROM dual;
```

– L'heure du jour formatée de la sorte :

```
--------------------------------
Il est : 13 heures et 07 minutes
```

```
SQL> SELECT 'Il est : '||TO_CHAR(SYSDATE,'HH24')||' heures et '||
  2                     TO_CHAR(SYSDATE,'MM')||' minutes' " "
  3  FROM DUAL;
```

– La date du jour, l'heure du jour et les secondes écoulées depuis minuit.

```
SQL> SELECT TO_CHAR(SYSDATE,'DD/MM/YYYY HH24:MI:SSSS')
  2  FROM dual;
```

– La date dans trois ans et dix mois.

```
SQL> SELECT TO_CHAR( SYSDATE,'DD/MM/YYYY'),
  2         TO_CHAR( SYSDATE + TO_YMINTERVAL('03-10'),
  3               'DD/MM/YYYY')
  4  FROM dual;
```

– Le nom, le prénom et le salaire des employés formatés de la manière suivante :

```
NOM        PRENOM     Salaire en €
---------- ---------- ------------------
Fuller     Andrew             10 000,00€
Callahan   Laura               2 000,00€
Peacock    Margaret            2 856,00€
Leverling  Janet               3 500,00€
Davolio    Nancy               3 135,00€
...
```

```
SQL> SELECT NOM,PRENOM,
  2         TO_CHAR( SALAIRE, '99G999D00U') "Salaire en €"
  3  FROM EMPLOYES;
```

Atelier 5.2 Les conversions SQL

Questions

1. Quelles sont les fonctions qui permettent en SQL de mettre en œuvre une structure conditionnelle de type `IF..THEN..ELSE` ?

   Réponse : DECODE, CASE

2. Laquelle de ces fonctions retourne la première expression `NOT NULL` de la liste des paramètres. ?

   A. `NULLIF`

   B. `COALESCE`

   C. `CASE`

   D. `LEAST`

   E. `GREATEST`

   F. `DECODE`

   Réponse : B

3. Pour les mêmes choix que la question précédente, quelle est la fonction permettant de renvoyer la plus petite valeur dans une liste de valeurs ?

   Réponse : D

Exercice n° 1 Les fonctions générales

Écrivez les requêtes permettant d'afficher :

– Le nom, le prénom, le salaire et la commission formatée de la sorte :

```
NOM                  PRENOM        SALAIRE Commission
-------------------- ---------- ---------- --------------------
Fuller               Andrew          10000 Pas de commission
Buchanan             Steven           8000 Pas de commission
Peacock              Margaret         2856 250
Leverling            Janet            3500 1000
Davolio              Nancy            3135 1500
Dodsworth            Anne             2180 0
King                 Robert           2356 800
Suyama               Michael          2534 600
Callahan             Laura            2000 Pas de commission
```

```
SQL> SELECT NOM, PRENOM, SALAIRE,
  2         DECODE( COMMISSION, NULL, 'Pas de commission',
  3                 COMMISSION) "Commission"
  4  FROM EMPLOYES;
```

– Le nom du produit, la plus grande valeur entre la valeur des produits en stock et la valeur des produits commandés pour tous les produits disponibles. La valeur du

stock ou de la commande est calculée en multipliant la plus grande valeur du stock ou de la commande par le prix unitaire. Toutes les valeurs des produits commandés doivent être affichées avec une valeur négative.

```
NOM_PRODUIT                              Valeur Stock
---------------------------------------- --------------------
Raclette Courdavault                              21.725,00€
Chai                                               3.510,00€
Chang                                             -3.800,00€
Aniseed Syrup                                     -3.500,00€
...
```

```
SQL> SELECT NOM_PRODUIT,
  2  TO_CHAR( DECODE(
  3               GREATEST( UNITES_STOCK, NVL(UNITES_COMMANDEES,0)),
  4               UNITES_COMMANDEES,
  5               -1*UNITES_COMMANDEES,
  6               UNITES_STOCK
  7               ) * PRIX_UNITAIRE, '99G999D00U') "Valeur Stock"
  8  FROM PRODUITS
  9  WHERE INDISPONIBLE <> -1;
```

ou

```
SQL> SQL> SELECT NOM_PRODUIT,
  2  TO_CHAR( CASE
  3              WHEN UNITES_STOCK > NVL(UNITES_COMMANDEES,0) THEN
  4                   UNITES_STOCK * PRIX_UNITAIRE
  5              ELSE -1 * UNITES_COMMANDEES * PRIX_UNITAIRE
  6           END , '99G999D00U') "Valeur Stock"
  7  FROM PRODUITS
  8  WHERE INDISPONIBLE <> -1;
```

– La société, l'adresse et le numéro de fax des fournisseurs. S'il n y a pas de numéro de fax renseigné, affichez le numéro de téléphone.

```
SQL> SELECT SOCIETE, ADRESSE, COALESCE(FAX,TELEPHONE)
  2  FROM FOURNISSEURS;
```

Atelier 6.1 Groupement des données

Questions

1. Quelle est la requête qui renvoie la valeur suivante 461 ?

   A.
   ```
   SELECT TRUNC(AVG(COMMISSION)) FROM EMPLOYES;
   ```

   B.
   ```
   SELECT TRUNC(AVG(COMMISSION)) FROM EMPLOYES
   WHERE COMMISSION IS NOT NULL;
   ```

   C.
   ```
   SELECT TRUNC(AVG(NVL(COMMISSION,0)))
   FROM EMPLOYES;
   ```

   D.
   ```
   SELECT TRUNC(NVL(AVG(COMMISSION),0))
   FROM EMPLOYES;
   ```

   Réponse : C

2. Quelle est la requête qui renvoie le nombre des employés qui ont saisie une commande ?

   A.
   ```
   SELECT COUNT( DISTINCT NO_EMPLOYE) FROM COMMANDES;
   ```

   B.
   ```
   SELECT COUNT( NO_EMPLOYE) FROM COMMANDES;
   ```

   C.
   ```
   SELECT DISTINCT COUNT( NO_EMPLOYE) FROM COMMANDES;
   ```

   Réponse : A

3. Quelle est la syntaxe du « **GROUP BY** » pour la requête suivante :

```
SELECT CODE_CLIENT,
       EXTRACT( YEAR FROM DATE_ENVOI),
       EXTRACT( MONTH FROM DATE_ENVOI),
       COUNT(NO_COMMANDE),
       SUM(PORT)
```

   A.
   ```
   CODE_CLIENT,
   EXTRACT( YEAR FROM DATE_ENVOI),
   EXTRACT( :MONTH FROM DATE_ENVOI)
   ```

   B.
   ```
   CODE_CLIENT, DATE_ENVOI
   ```

   C.
   ```
   CODE_CLIENT,EXTRACT( YEAR FROM DATE_ENVOI)
   ```

   D.
   ```
   CODE_CLIENT,
   TO_CHAR( DATE_ENVOI,'YYYY'),
   TO_CHAR( DATE_ENVOI,'MM')
   ```

   Réponse : A

Exercice n° 1 Les fonctions d'agrégat

Écrivez les requêtes permettant d'afficher :

– La valeur totale des produits en stock et la valeur totale des produits commandés.

```
SQL> SELECT SUM(PRIX_UNITAIRE*UNITES_STOCK),
  2         SUM(PRIX_UNITAIRE*UNITES_COMMANDEES)
  3  FROM PRODUITS;
```

– La valeur totale des produits vendus et le total du chiffre d'affaire, la valeur totale des produits vendus moins la remise. Le champ REMISE représente un pourcentage de remise.

```
SQL> SELECT SUM(PRIX_UNITAIRE*QUANTITE),
  2         SUM(PRIX_UNITAIRE*QUANTITE*( 1 - REMISE))
  3  FROM DETAILS_COMMANDES;
```

– La masse salariale.

```
SQL> SELECT SUM(SALAIRE+NVL(COMMISSION,0)) FROM EMPLOYES;
```

Exercice n° 2 Le groupement des données

Écrivez les requêtes permettant d'afficher :

– La masse salariale pour chaque fonction des employés.

```
SQL> SELECT FONCTION, SUM(SALAIRE+NVL(COMMISSION,0)) FROM EMPLOYES
  2  GROUP BY FONCTION;
```

– Le nombre des commandes et la somme des frais de port pour chaque client et par année et par mois.

```
SQL> SELECT CODE_CLIENT,
  2         EXTRACT( YEAR FROM DATE_ENVOI),
  3         EXTRACT( MONTH FROM DATE_ENVOI),
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY CODE_CLIENT,
  8           EXTRACT( YEAR FROM DATE_ENVOI),
  9           EXTRACT( MONTH FROM DATE_ENVOI);
```

ou

```
SQL> SELECT CODE_CLIENT,
  2         ANNEE,
  3         MOIS,
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY CODE_CLIENT,
  8           ANNEE,
  9           MOIS;
```

– La somme totale des produits en stock et la somme totale des produits commandés par fournisseur et par catégorie des produits.

```
SQL> SELECT NO_FOURNISSEUR,CODE_CATEGORIE,
  2         SUM(PRIX_UNITAIRE*UNITES_STOCK),
  3         SUM(PRIX_UNITAIRE*UNITES_COMMANDEES)
  4  FROM PRODUITS
  5  GROUP BY NO_FOURNISSEUR,
  6           CODE_CATEGORIE;
```

Exercice n° 3 La sélection de groupe

Écrivez les requêtes permettant d'afficher :

– La somme des produits en stock et la somme des produits commandés pour les fournisseurs qui ont un numéro compris entre 3 et 6 et qui vendent au moins trois catégories de produits.

```
SQL> SELECT NO_FOURNISSEUR, SUM(PRIX_UNITAIRE*UNITES_STOCK),
  2         SUM(PRIX_UNITAIRE*UNITES_COMMANDEES)
  3  FROM PRODUITS
  4  WHERE NO_FOURNISSEUR BETWEEN 3 AND 6
  5  GROUP BY NO_FOURNISSEUR
  6  HAVING COUNT(DISTINCT CODE_CATEGORIE) >= 3 ;
```

– La somme totale des produits vendus et la somme du chiffre d'affaire pour les commandes qui comportent plus de 50 produits.

```
SQL> SELECT NO_COMMANDE,
  2         SUM(PRIX_UNITAIRE*QUANTITE)
  3  FROM DETAILS_COMMANDES
  4  GROUP BY NO_COMMANDE
  5  HAVING COUNT(DISTINCT REF_PRODUIT) > 50 ;
```

– Le nombre des commandes et la somme des frais de port pour chaque client et par année et par mois. Il faut afficher uniquement les clients qui ont commandé plus de quinze fois dans le mois et dont leur frais de port dans le mois sont supérieurs à 1400€.

```
SQL> SELECT CODE_CLIENT,
  2         EXTRACT( YEAR FROM DATE_ENVOI),
  3         EXTRACT( MONTH FROM DATE_ENVOI),
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY CODE_CLIENT,
  8           EXTRACT( YEAR FROM DATE_ENVOI),
  9           EXTRACT( MONTH FROM DATE_ENVOI)
 10  HAVING COUNT(NO_COMMANDE) > 15
 11     AND SUM(PORT) > 1400;
```

Atelier 6.2 Agrégation et Analyse

Questions

1. Quelle est l'extension de la clause « **GROUP BY** » qui vous permet de générer des sous-totaux pour les attributs spécifiés ?

   Réponse : ROLLUP

2. Quelle est l'extension de la clause « **GROUP BY** » qui vous permet de définir plusieurs groupes dans la même requête. ?

   Réponse : GROUPING SETS

3. Quelles sont les syntaxes incorrectes ?

   A. `...GROUP BY GROUPING SETS(ROLLUP(...`

   B. `...GROUP BY CUBE( GROUPING SETS(...`

   C. `...GROUP BY ROLLUP( GROUPING SETS...`

   D. `...GROUP BY GROUPING SETS(CUBE(...`

   E. `...CUBE( GROUP BY...`

   Réponse : B, C, E

Exercice n° 1 Les extensions ROLLUP et CUBE

Écrivez les requêtes permettant d'afficher :

– Le nombre des commandes et la somme des frais de port pour chaque client, par année et par mois. Afficher également la somme des commandes et la somme des frais de port pour chaque client par année et la somme totale.

```
SQL> SELECT CODE_CLIENT,
  2         EXTRACT( YEAR FROM DATE_COMMANDE) "Année",
  3         EXTRACT( MONTH FROM DATE_COMMANDE)"Mois",
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY CODE_CLIENT,
  8           EXTRACT( YEAR FROM DATE_COMMANDE),
  9           ROLLUP ( EXTRACT( MONTH FROM DATE_COMMANDE));
```

ou

```
SQL> SELECT CODE_CLIENT,
  2         ANNEE,
  3         MOIS,
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY CODE_CLIENT, ANNEE,
  8  ROLLUP (MOIS);
```

– Les mêmes informations que la requête précédente mais cette fois-ci affichez les totaux par client, par année ainsi que le total global.

```
SQL> SELECT CODE_CLIENT,
  2         ANNEE,
  3         MOIS,
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY ROLLUP ( CODE_CLIENT, ANNEE, MOIS);
```

– Les mêmes informations que la requête précédente mais cette fois-ci affichez tous les totaux possibles.

```
SQL> SELECT CODE_CLIENT,
  2         ANNEE,
  3         MOIS,
  4         COUNT(NO_COMMANDE),
  5         SUM(PORT)
  6  FROM COMMANDES
  7  GROUP BY CUBE ( CODE_CLIENT, ANNEE, MOIS);
```

– La somme des produits en stock et la somme des produits commandés pour chaque catégorie et par fournisseur. Afficher également les totaux par catégorie de produits.

```
SQL> SELECT CODE_CATEGORIE,
  2         NO_FOURNISSEUR,
  3         SUM(PRIX_UNITAIRE*UNITES_STOCK),
  4         SUM(PRIX_UNITAIRE*NVL(UNITES_COMMANDEES,0))
  5  FROM PRODUITS
  6  WHERE INDISPONIBLE = 0
  7  GROUP BY CODE_CATEGORIE,ROLLUP( NO_FOURNISSEUR);
```

Exercice n° 2 L'extension GROUPING SETS

Écrivez les requêtes permettant d'afficher :

– La somme des produits en stock et la somme des produits commandés pour chaque catégorie. Dans la même requête affichez également les mêmes sommes mais pour chaque fournisseur.

```
SQL> SELECT CODE_CATEGORIE,
  2         NO_FOURNISSEUR,
  3         SUM(PRIX_UNITAIRE*UNITES_STOCK),
  4         SUM(PRIX_UNITAIRE*NVL(UNITES_COMMANDEES,0))
  5  FROM PRODUITS
  6  WHERE INDISPONIBLE = 0
  7  GROUP BY GROUPING SETS(CODE_CATEGORIE,NO_FOURNISSEUR)
  8  ORDER BY CODE_CATEGORIE,NO_FOURNISSEUR;
```

– Le nombre des commandes et la somme des frais de port saisis par un employé pour chaque client. Dans la même requête affichez également les mêmes sommes mais par année et par mois. Afficher également les totaux par employé et par année et les totaux globaux.

```
SQL> SELECT NO_EMPLOYE,
  2         CODE_CLIENT,
  3         ANNEE,
  4         MOIS ,
  5         COUNT(NO_COMMANDE),
  6         SUM(PORT)
```

```
  7  FROM COMMANDES
  8  GROUP BY GROUPING SETS (
  9              ROLLUP ( NO_EMPLOYE,CODE_CLIENT),
 10              ROLLUP ( ANNEE,MOIS))
 11  HAVING GROUP_ID() = 0
 12  ORDER BY NO_EMPLOYE,
 13           CODE_CLIENT,
 14           ANNEE,
 15           MOIS;
```

– La somme des frais de port saisis par un employé pour chaque client, par année et par mois. Afficher également la somme des commandes et la somme des frais de port pour chaque client par année et la somme totale formatées de la sorte :

```
Employé       Client      Année   Mois           Port
------------- ----------- ------- ------- ----------
   1          AROUT       2010    01            97,5
   1          AROUT       2010    02            78,8
   1          AROUT       2010    03           182,4
   1          AROUT       2010    05              54
   1          AROUT       2010    07           160,4
   1          AROUT       2010    08            52,9
   1          AROUT       2010    09           279,3
   1          AROUT       2010    10           259,4
   1          AROUT       2010    11             153
   1          AROUT       2010    *******     1317,7
...
   1          AROUT       2011    06            559,4
   1          AROUT       2011    *******     1708,8
   1          AROUT       ******* *******     3026,5
...
 111          VINET       2010    02              219
 111          VINET       2010    06            275,1
...
 111          VINET       2011    06              382
 111          VINET       2011    *******     1638,9
 111          VINET       ******* *******     2744,4
 111          ********** ******* *******    32587,2
************* ********** ******* *******  1007453,5
```

```
SQL> SELECT
  2    CASE GROUPING( NO_EMPLOYE ) WHEN 0 THEN
  3    TO_CHAR(NO_EMPLOYE,'999') ELSE '*************' END "Employé",
  4    CASE GROUPING( CODE_CLIENT ) WHEN 0 THEN
  5    CODE_CLIENT ELSE '************' END "Client",
  6    CASE GROUPING( TO_CHAR( DATE_COMMANDE, 'YYYY') ) WHEN 0 THEN
  7    TO_CHAR( DATE_COMMANDE, 'YYYY') ELSE '*******' END "Année",
  8    CASE GROUPING( TO_CHAR( DATE_COMMANDE, 'MM') ) WHEN 0 THEN
  9    TO_CHAR( DATE_COMMANDE, 'MM') ELSE '*******' END "Mois",
 10    SUM(PORT) "Port"
 11  FROM COMMANDES
 12  GROUP BY ROLLUP( NO_EMPLOYE, CODE_CLIENT,
 13                   TO_CHAR( DATE_COMMANDE, 'YYYY'),
 14                   TO_CHAR( DATE_COMMANDE, 'MM'))
 15  ORDER BY NO_EMPLOYE, CODE_CLIENT,
```

```
 16           TO_CHAR( DATE_COMMANDE, 'YYYY'),
 17           TO_CHAR( DATE_COMMANDE, 'MM');
```

ou

```
SQL> SELECT
  2    CASE GROUPING( NO_EMPLOYE ) WHEN 0 THEN
  3    TO_CHAR(NO_EMPLOYE,'999') ELSE '**************' END "Employé",
  4    CASE GROUPING( CODE_CLIENT ) WHEN 0 THEN
  5    CODE_CLIENT ELSE '***********' END "Client",
  6    CASE GROUPING( ANNEE ) WHEN 0 THEN
  7    TO_CHAR(ANNEE) ELSE '*******' END "Année",
  8    CASE GROUPING( MOIS ) WHEN 0 THEN
  9    TO_CHAR(MOIS) ELSE '*******' END "Mois",
 10    SUM(PORT) "Port"
 11  FROM COMMANDES
 12  GROUP BY ROLLUP( NO_EMPLOYE, CODE_CLIENT,
 13                   ANNEE,
 14                   MOIS)
 15  ORDER BY NO_EMPLOYE, CODE_CLIENT,
 16           ANNEE,
 17           MOIS;
```

Atelier 7.1 Les requêtes multi-tables

Questions

1. Sachant que la table COMMANDES a 830 enregistrements et la table DETAILS_COMMANDES a 2155 enregistrements, combien d'enregistrements retournent une requête qui interroge ces deux tables sans aucune jointure ?

   Réponse : 1 788 650

2. Quelles sont les syntaxes incorrectes ?

   A.
   ```
   SELECT COUNT(DISTINCT A.NO_EMPLOYE)
   FROM EMPLOYES A, COMMANDES B
   WHERE A.NO_EMPLOYE = B.NO_EMPLOYE (+);
   ```

   B.
   ```
   SELECT COUNT(DISTINCT A.NO_EMPLOYE)
   FROM EMPLOYES A, COMMANDES B
   WHERE A.NO_EMPLOYE = (+)B.NO_EMPLOYE;
   ```

   C.
   ```
   SELECT COUNT(DISTINCT A.NO_EMPLOYE)
   FROM EMPLOYES A, COMMANDES B
   WHERE A.NO_EMPLOYE(+)= B.NO_EMPLOYE ;
   ```

   D.
   ```
   SELECT COUNT(DISTINCT A.NO_EMPLOYE)
   FROM EMPLOYES A, COMMANDES B
   WHERE (+)A.NO_EMPLOYE = B.NO_EMPLOYE (+);
   ```

   E.
   ```
   SELECT COUNT(DISTINCT A.NO_EMPLOYE)
   FROM EMPLOYES A, COMMANDES B
   WHERE A.NO_EMPLOYE = B.NO_EMPLOYE;
   ```

   Réponse : B, C, D

3. Quelle est la requête dans la liste précédente qui retrouve tous les employés même s'ils n'ont pas passé de commandes ?

   Réponse : A

4. Quelles sont les syntaxes incorrectes ?

   A.
   ```
   SELECT COUNT(*) FROM EMPLOYES A
   OUTER JOIN EMPLOYES B ;
   ```

   B.
   ```
   SELECT COUNT(*) FROM EMPLOYES A
   LEFT OUTER JOIN EMPLOYES B ;
   ```

   C.
   ```
   SELECT COUNT(*) FROM EMPLOYES A
   NATURAL JOIN EMPLOYES B ;
   ```

   D.
   ```
   SELECT COUNT(*) FROM EMPLOYES A JOIN EMPLOYES B
   ON ( A.NO_EMPLOYE = B.REND_COMPTE );
   ```

   E.
   ```
   SELECT COUNT(*) FROM EMPLOYES A LEFT OUTER JOIN
   EMPLOYES B ON ( A.NO_EMPLOYE = B.REND_COMPTE );
   ```

   F.
   ```
   SELECT COUNT(*) FROM EMPLOYES A
   JOIN EMPLOYES B USING ( NO_EMPLOYE );
   ```

   Réponse : A, B

5. Quelle est la requête dans la liste précédente qui retrouve tous les employés et les employés gérés par eux si tél est le cas sinon « **NULL** » ?

Réponse : E

Exercice n° 1 Les équijointures

Écrivez les requêtes utilisant la syntaxe de jointure relationnelle (basée sur la norme ANSI SQL/86), ainsi que la deuxième syntaxe qui respecte la norme AINSI SQL/92.

– Le nom, le prénom et la société cliente pour les employés qui ont effectué une vente pour les clients de Paris.

```
1.
SQL> SELECT DISTINCT NOM, PRENOM, SOCIETE
  2  FROM EMPLOYES A, COMMANDES B, CLIENTS C
  3  WHERE A.NO_EMPLOYE  = B.NO_EMPLOYE  AND
  4        B.CODE_CLIENT = C.CODE_CLIENT AND
  5        C.VILLE       = 'Paris';
2.
SQL> SELECT NOM, PRENOM, SOCIETE,
  2         TO_CHAR( DATE_COMMANDE,'FMDD Month YYYY'),
  3         TO_CHAR( PORT,'9G990D00U')
  4  FROM CLIENTS JOIN COMMANDES  USING ( CODE_CLIENT )
  5               JOIN EMPLOYES   USING ( NO_EMPLOYE  )
  6  WHERE VILLE       = 'Paris';
```

– La société cliente, le nombre des produits commandés, la ville et le pays qui ont commandé plus de 115 références de produits au mois de mai 2011.

```
1.
SQL> SELECT SOCIETE, COUNT(DISTINCT C.REF_PRODUIT),
  2         VILLE, PAYS
  3  FROM CLIENTS A, COMMANDES B, DETAILS_COMMANDES C
  4  WHERE A.CODE_CLIENT = B.CODE_CLIENT
  5    AND B.NO_COMMANDE = C.NO_COMMANDE
  6    AND B.ANNEE       = 2011
  7    AND B.MOIS        = 5
  8  GROUP BY SOCIETE, VILLE, PAYS
  9  HAVING COUNT(DISTINCT C.REF_PRODUIT) > 115;
2.
SQL> SELECT SOCIETE, COUNT(DISTINCT REF_PRODUIT), VILLE, PAYS
  2  FROM CLIENTS JOIN COMMANDES USING (CODE_CLIENT )
  3               JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  4  WHERE ANNEE  = 2011
  5    AND MOIS   = 5
  6  GROUP BY SOCIETE, VILLE, PAYS
  7  HAVING COUNT(DISTINCT REF_PRODUIT) > 115;
```

– Le nom de la catégorie du produit, la société fournisseur et le nom du produit, uniquement pour les produits des catégories 1, 4 et 7.

```
1.
SQL> SELECT NOM_CATEGORIE, SOCIETE, NOM_PRODUIT
  2  FROM PRODUITS A, FOURNISSEURS B, CATEGORIES C
  3  WHERE A.NO_FOURNISSEUR = B.NO_FOURNISSEUR AND
  4        A.CODE_CATEGORIE = C.CODE_CATEGORIE AND
  5        A.CODE_CATEGORIE IN (1,4,7);
```

```
2.
SQL> SELECT NOM_CATEGORIE, SOCIETE, NOM_PRODUIT
  2  FROM PRODUITS JOIN FOURNISSEURS USING ( NO_FOURNISSEUR )
  3                JOIN CATEGORIES   USING ( CODE_CATEGORIE )
  4  WHERE CODE_CATEGORIE IN (1,4,7);
```

– La société cliente, la société fournisseur et leur ville pour les clients qui sont localisés dans une ville d'un fournisseur (Il s'agit d'une jointure entre la table `CLIENTS` et `FOURNISSEURS`).

```
1.
SQL> SELECT A.SOCIETE, B.SOCIETE, A.VILLE
  2  FROM FOURNISSEURS A, CLIENTS B
  3  WHERE A.VILLE = B.VILLE;
2.
SQL> SELECT A.SOCIETE, B.SOCIETE, VILLE
  2  FROM FOURNISSEURS A JOIN CLIENTS B USING ( VILLE );
```

– Les sociétés clientes qui ont commandé le produit `'Chai'`.

```
1.
SQL> SELECT SOCIETE, VILLE
  2  FROM CLIENTS A, COMMANDES B, DETAILS_COMMANDES C, PRODUITS D
  3  WHERE A.CODE_CLIENT = B.CODE_CLIENT AND
  4        B.NO_COMMANDE = C.NO_COMMANDE AND
  5        C.REF_PRODUIT = D.REF_PRODUIT AND
  6        D.NOM_PRODUIT = 'Chai';
2.
SQL> SELECT SOCIETE, VILLE
  2  FROM CLIENTS JOIN COMMANDES         USING ( CODE_CLIENT )
  3               JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  4               JOIN PRODUITS          USING ( REF_PRODUIT )
  5  WHERE NOM_PRODUIT = 'Chai';
```

Exercice n° 2 Les jointures externes et autojointures

– Tous les clients et le cumul des quantités vendues pour les clients qui ont passé des commandes. Affichez les enregistrements par ordre décroissant de cumul des commandes avec les valeurs « **NULL** » à la fin.

```
1.
SQL> SELECT SOCIETE, SUM(C.QUANTITE)
  2  FROM CLIENTS A, COMMANDES B, DETAILS_COMMANDES C
  3  WHERE A.CODE_CLIENT = B.CODE_CLIENT(+) AND
  4        B.NO_COMMANDE = C.NO_COMMANDE(+)
  5  GROUP BY SOCIETE
  6  ORDER BY SUM(C.QUANTITE) DESC NULLS LAST;
2.
SQL> SELECT SOCIETE, SUM(QUANTITE)
  2  FROM CLIENTS LEFT OUTER JOIN COMMANDES USING ( CODE_CLIENT )
  3       LEFT OUTER JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  4  GROUP BY SOCIETE
  5  ORDER BY SUM(QUANTITE) DESC NULLS LAST;
```

– Les localités des clients et le cumul des quantités vendues par localité. Affichez les enregistrements par ordre décroissant de cumul des commandes avec les valeurs « **NULL** » à la fin.

```
1.
SQL> SELECT VILLE, SUM(C.QUANTITE)
  2  FROM CLIENTS A, COMMANDES B, DETAILS_COMMANDES C
  3  WHERE A.CODE_CLIENT = B.CODE_CLIENT(+) AND
  4        B.NO_COMMANDE = C.NO_COMMANDE(+)
  5  GROUP BY VILLE
  6  ORDER BY SUM(C.QUANTITE) DESC NULLS LAST;
2.
SQL> SELECT VILLE, SUM(QUANTITE)
  2  FROM CLIENTS LEFT OUTER JOIN COMMANDES USING ( CODE_CLIENT )
  3       LEFT OUTER JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  4  GROUP BY VILLE
  5  ORDER BY SUM(QUANTITE) DESC NULLS LAST;
```

– Le nom, le prénom, la fonction de tous les employés, la somme des frais de port et le cumul des ventes (prix unitaire fois la quantité) pour les employés qui ont passé des commandes. Affichez les enregistrements par ordre décroissant de cumul des ventes avec les valeurs « **NULL** » à la fin.

```
1.
SQL> SELECT NOM, PRENOM,
  2         TO_CHAR( SUM(PORT),'999G999D00U'),
  3         TO_CHAR( SUM(QUANTITE*PRIX_UNITAIRE),'999G999G999D00U')
  4  FROM EMPLOYES A, COMMANDES B, DETAILS_COMMANDES C
  5  WHERE A.NO_EMPLOYE = B.NO_EMPLOYE (+) AND
  6        B.NO_COMMANDE= C.NO_COMMANDE(+)
  7  GROUP BY NOM, PRENOM
  8  ORDER BY 4 DESC NULLS LAST;
2.
SQL> SELECT NOM, PRENOM,
  2         TO_CHAR( SUM(PORT),'999G999D00U'),
  3         TO_CHAR( SUM(QUANTITE*PRIX_UNITAIRE),'999G999G999D00U')
  4  FROM EMPLOYES LEFT OUTER JOIN COMMANDES USING ( NO_EMPLOYE )
  5       LEFT OUTER JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  6  GROUP BY NOM, PRENOM
  7  ORDER BY 4 DESC NULLS LAST;
```

– Le nom, le prénom, la fonction des supérieurs hiérarchiques ainsi le nom et prénom des employés gérés par eux.

```
1.
SQL> SELECT MGR.NOM, MGR.PRENOM, MGR.FONCTION, EMP.NOM, EMP.PRENOM
  2  FROM EMPLOYES MGR, EMPLOYES EMP
  3  WHERE MGR.NO_EMPLOYE = EMP.REND_COMPTE;
2.
SQL> SELECT MGR.NOM, MGR.PRENOM, MGR.FONCTION, EMP.NOM, EMP.PRENOM
  2  FROM EMPLOYES MGR JOIN EMPLOYES EMP
  3       ON( MGR.NO_EMPLOYE = EMP.REND_COMPTE);
```

– Le nom, le prénom, la fonction des tous les employés, le nom et prénom des employés gérés par eux, si tél est le cas, ainsi que le nom et prénom des employés gérés par les précédents si tél est le cas.

```
1.
SQL> SELECT A.NOM||' '||A.PRENOM "Employé", A.FONCTION,
  2         B.NOM||' '||B.PRENOM "Gérés par A",
  3         C.NOM||' '||C.PRENOM "Gérés par B"
  4  FROM EMPLOYES A, EMPLOYES B, EMPLOYES C
  5  WHERE A.NO_EMPLOYE = B.REND_COMPTE (+) AND
```

```
  6        B.NO_EMPLOYE = C.REND_COMPTE (+);
2.
SQL> SELECT A.NOM||' '||A.PRENOM "Employé", A.FONCTION,
  2         B.NOM||' '||B.PRENOM "Gérés par A",
  3         C.NOM||' '||C.PRENOM "Gérés par B"
  4  FROM EMPLOYES A LEFT OUTER JOIN EMPLOYES B
  5           ON ( A.NO_EMPLOYE = B.REND_COMPTE )
  6       LEFT OUTER JOIN EMPLOYES C
  7           ON ( B.NO_EMPLOYE = C.REND_COMPTE );
```

Atelier 8.1 Les jointures complexes

Questions

1. Quel est l'opérateur qui force un ordre de tri des enregistrements ?

   A. `MINUS`

   B. `UNION DISTINCT`

   C. `UNION ALL`

   D. `INTERSECT`

   Réponse : B

2. Quel est l'opérateur dans la liste précédente qui n'est pas commutative ?

   Réponse : A

3. Quel est l'option par défaut pour l'opérateur « **`UNION`** » ?

   Réponse : `DISTINCT`

Exercice n° 1 Les opérateurs ensemblistes

Écrivez les requêtes permettant d'afficher :

– Pour un mailing, il faut trouver l'ensemble des tiers de l'entreprise (les sociétés clientes ou fournisseurs) ainsi que leur adresse et ville de résidence.

```
SQL> SELECT SOCIETE, ADRESSE, VILLE
  2  FROM FOURNISSEURS
  3  UNION
  4  SELECT SOCIETE, ADRESSE, VILLE
  5  FROM CLIENTS;
```

– Toutes les commandes qui comportent en même temps des produits de catégorie 1 du fournisseur 1 et produits de catégorie 2 du fournisseur 2.

```
SQL> SELECT NO_COMMANDE
  2  FROM  DETAILS_COMMANDES
  3  WHERE REF_PRODUIT IN ( SELECT REF_PRODUIT FROM PRODUITS
  4                         WHERE  CODE_CATEGORIE = 1 AND
  5                                NO_FOURNISSEUR = 1)
  6  INTERSECT
  7  SELECT NO_COMMANDE
  8  FROM  DETAILS_COMMANDES
  9  WHERE REF_PRODUIT IN ( SELECT REF_PRODUIT FROM PRODUITS
 10                         WHERE  CODE_CATEGORIE = 2 AND
 11                                NO_FOURNISSEUR = 2);
```

– Les produits qu'on ne commande qu'à Paris.

```
SQL> SELECT REF_PRODUIT
  2  FROM  PRODUITS
  3  MINUS
  4  SELECT REF_PRODUIT
  5  FROM  CLIENTS JOIN COMMANDES USING ( CODE_CLIENT)
```

```
  6                JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  7  WHERE VILLE = 'Paris';
```

– Les sociétés clientes qui ont commandé le produit `'Chai'` mais également qui ont commandé plus de vingt cinq produits.

```
SQL> SELECT DISTINCT SOCIETE
  2  FROM CLIENTS JOIN COMMANDES USING (CODE_CLIENT )
  3               JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  4  GROUP BY SOCIETE
  5  HAVING COUNT(DISTINCT REF_PRODUIT) > 25
  6  INTERSECT
  7  SELECT DISTINCT SOCIETE
  8  FROM CLIENTS JOIN COMMANDES         USING ( CODE_CLIENT )
  9               JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
 10               JOIN PRODUITS          USING ( REF_PRODUIT )
 11  WHERE NOM_PRODUIT = 'Chai';
```

Atelier 8.2 Les jointures complexes

Questions

1. Quels sont les clauses qui ne peuvent pas comporter des sous-requêtes ?

   A. SELECT

   B. FROM

   C. WHERE

   D. GROUP BY

   E. ORDER BY

   F. HAVING

   Réponse : D, E

2. Quels sont les opérateurs logiques qui peuvent travailler avec des sous-requêtes multilignes ?

   A. =

   B. IN

   C. >

   D. <

   E. >=

   F. <=

   G. !=

   H. ALL=

   I. ANY<

   J. LIKE

   Réponse : B, H, I

Exercice n° 1 Les sous-requêtes monolignes

Écrivez les requêtes permettant d'afficher :

– Les produits pour lesquels la quantité en stock est inférieure à la moyenne.

```
SQL> SELECT * FROM PRODUITS
  2  WHERE UNITES_STOCK < ( SELECT AVG(UNITES_STOCK) FROM PRODUITS);
```

– Les sociétés clientes et les numéros des commandes qui ont un nombre égal ou supérieur de produits achetés que la commande numéro '216798'.

```
SQL> SELECT SOCIETE, NO_COMMANDE, COUNT( REF_PRODUIT)
  2  FROM CLIENTS NATURAL JOIN COMMANDES
  3                 JOIN DETAILS_COMMANDES USING(NO_COMMANDE)
  4  GROUP BY SOCIETE, NO_COMMANDE
  5  HAVING COUNT( REF_PRODUIT) >= ( SELECT COUNT( REF_PRODUIT)
  6                                  FROM   COMMANDES  NATURAL JOIN
```

```
  7                                          DETAILS_COMMANDES
  8                                  WHERE  NO_COMMANDE = 216798);
```

Exercice n° 2 Les sous-requêtes multilignes

Écrivez les requêtes permettant d'afficher :

– Les sociétés clientes et leurs commandes pour tous les produits livrés par un fournisseur qui habite Paris.

```
SQL> SELECT SOCIETE, NO_COMMANDE
  2  FROM CLIENTS NATURAL JOIN COMMANDES
  3                 JOIN DETAILS_COMMANDES USING(NO_COMMANDE)
  4  WHERE REF_PRODUIT IN ( SELECT REF_PRODUIT
  5                         FROM PRODUITS NATURAL JOIN FOURNISSEURS
  6                         WHERE  VILLE = 'Paris');
```

– Les sociétés clientes françaises qui ont commandé le produit `'Chai'` mais également qui ont commandé plus de 39 produits le `'01/05/2011'`.

```
SQL> SELECT SOCIETE, NO_COMMANDE
  1  SELECT SOCIETE, NO_COMMANDE
  2  FROM CLIENTS JOIN COMMANDES         USING ( CODE_CLIENT )
  3               JOIN DETAILS_COMMANDES USING ( NO_COMMANDE )
  4               JOIN PRODUITS          USING ( REF_PRODUIT )
  5  WHERE NOM_PRODUIT = 'Chai'
  6    AND PAYS        = 'France'
  7    AND CODE_CLIENT IN (SELECT CODE_CLIENT
  8                        FROM CLIENTS JOIN COMMANDES
  9                                    USING (CODE_CLIENT )
 10                             JOIN DETAILS_COMMANDES
 11                                    USING ( NO_COMMANDE )
 12                      WHERE DATE_COMMANDE = '01/05/2011'
 13                      GROUP BY CODE_CLIENT
 14                      HAVING COUNT( DISTINCT REF_PRODUIT) >= 40);
```

Exercice n° 3 Les sous-requêtes un tableau

Écrivez les requêtes permettant d'afficher :

– Le nom, prénom, la fonction, le nom du produit et la date de la commande pour les produits achetés par les clients qui habitent dans la même ville que le fournisseur.

```
SQL> SELECT NOM, PRENOM , FONCTION, NOM_PRODUIT, DATE_COMMANDE
  2  FROM EMPLOYES JOIN COMMANDES USING( NO_EMPLOYE )
  3                JOIN DETAILS_COMMANDES USING( NO_COMMANDE )
  4                JOIN PRODUITS USING( REF_PRODUIT)
  5  WHERE ( CODE_CLIENT, NO_FOURNISSEUR) IN (
  6   SELECT CODE_CLIENT, NO_FOURNISSEUR
  7   FROM CLIENTS JOIN FOURNISSEURS USING(VILLE));
```

Exercice n° 4 Les sous-requêtes synchronisée

Écrivez les requêtes permettant d'afficher :

– Les clients pour lesquels les frais de ports par commande dépassent leur moyenne globale des frais de ports.

```
SQL> SELECT SOCIETE FROM COMMANDES A, CLIENTS
  2  WHERE A.CODE_CLIENT = CLIENTS.CODE_CLIENT AND
  3        PORT > ( SELECT AVG(PORT) FROM COMMANDES B
  4                 WHERE A.CODE_CLIENT = B.CODE_CLIENT );
```

– Le produit, le fournisseur et les unités en stock pour les produits qui ont un stock inférieur à la moyenne des unités en stock pour les produits du même fournisseur.

```
SQL> SELECT NOM_PRODUIT, NO_FOURNISSEUR, UNITES_STOCK
  2  FROM PRODUITS A
  3  WHERE UNITES_STOCK >(SELECT AVG(UNITES_STOCK)
  4                       FROM PRODUITS B
  5                       WHERE A.NO_FOURNISSEUR =B.NO_FOURNISSEUR);
```

– Les clients et ses commandes pour les clients qui payent un port supérieur à la moyenne des commandes pour la même année.

```
SQL> SELECT CODE_CLIENT, NO_COMMANDE FROM COMMANDES A
  2  WHERE PORT > ( SELECT AVG(PORT) FROM COMMANDES B
  3                 WHERE A.ANNEE = B.ANNEE);
```

– Les employés avec leur salaire et le pourcentage correspondant par rapport au total de la masse salariale par fonction. Essayez d'utiliser une sous-requête dans la clause « **FROM** ».

```
SQL> SELECT NOM, SALAIRE, TO_CHAR( 100*SALAIRE/SUM_S, '999D00')||'%'
  2  "% total la fonction"
  3  FROM EMPLOYES, ( SELECT FONCTION, SUM(SALAIRE) SUM_S
  4                   FROM EMPLOYES
  5                   GROUP BY FONCTION ) SUM_EMPLOYES
  6  WHERE  EMPLOYES.FONCTION = SUM_EMPLOYES.FONCTION;
```

Atelier 9.1 Les fonctions analytiques

Questions

1. Quelles sont les fonctions analytiques ?

   Réponse : Vous pouvez utiliser n'importe quelle fonction d'agrégat « **SUM** », « **AVG** », « **COUNT** » … comme des fonctions analytiques avec l'indicateur « **OVER** ».

2. Quelle clause est une clause de fenêtrage implicite. ?

   Réponse : Si la clause « **ORDER BY** » est utilisée et qu'une clause de fenêtrage n'est pas spécifiée, une clause de fenêtrage implicite est appliquée « **UNBOUNDED PRECEDING** ».

3. Quelles sont les syntaxes incorrectes ?

   A.
   ```
   SELECT NOM, SUM(SALAIRE)FROM EMPLOYES;
   ```

   B.
   ```
   SELECT NOM, SUM(SALAIRE) OVER () FROM EMPLOYES;
   ```

   C.
   ```
   SELECT NOM, SUM(SALAIRE)
   OVER (PARTITION BY FONCTION ) FROM EMPLOYES;
   ```

   D.
   ```
   SELECT NOM, SUM(SALAIRE)
   OVER (PARTITION BY FONCTION ORDER BY NOM )
   FROM EMPLOYES;
   ```

   E.
   ```
   SELECT NOM, SUM(SALAIRE)
   OVER (ORDER BY NOM ) FROM EMPLOYES;
   ```

   F.
   ```
   SELECT NOM,  OVER (SUM(SALAIRE)) FROM EMPLOYES;
   ```

   G.
   ```
   SELECT NOM, SUM(SALAIRE)
   OVER (ORDER BY NOM PARTITION BY FONCTION )
   FROM EMPLOYES;
   ```

   Réponse : A, F, G

Exercice n° 1 Le partitionnement

Écrivez les requêtes permettant d'afficher :

– Le client, les commandes, les frais de port, le nombre des commandes du client dans le mois, la moyenne des frais de port pour le mois en cours et la moyenne des frais de port pour l'année.

```
SQL> col "Frais Port"      for 99g999D00
SQL> col "Avg Mois"        for 99g999D00
SQL> col "Avg Année"       for 99g999D00
SQL> col "Nombre Contrats" for 99g999D00
SQL> SELECT CODE_CLIENT,
  2    NO_COMMANDE,
  3    PORT "Frais Port",
  4    AVG(PORT) OVER(PARTITION BY MOIS) "Avg Mois",
  5    AVG(PORT) OVER(PARTITION BY ANNEE) "Avg Année",
  6    COUNT(NO_COMMANDE)OVER
```

```
  7       (PARTITION BY CODE_CLIENT, MOIS) "Nombre Contrats"
  8  FROM COMMANDES
  9  ORDER BY CODE_CLIENT, NO_COMMANDE, DATE_COMMANDE;
```

– Le nom, la fonction, le salaire, le poids du salaire dans la masse salariale de la fonction (le salaire divisé par la somme de tous les salaires), le poids du salaire dans la masse salariale de l'entreprise.

```
SQL> col "Salaire" for 999G999D00U
SQL> col "% Fonction" for 999D00U
SQL> col "% Total" for 999D00U
SQL> SELECT NOM,
  2        FONCTION,
  3        SALAIRE "Salaire",
  4        SALAIRE*100/SUM(SALAIRE)
  5                OVER( PARTITION BY FONCTION) "% Fonction",
  6        SALAIRE*100/SUM(SALAIRE) OVER() "% Total"
  7  FROM EMPLOYES;
```

– Le nom du produit, le fournisseur, la catégorie, les unités en stock, la moyenne des unités en stock pour le même fournisseur, la moyenne des unités en stock pour la même catégorie et somme totale des unités en stock.

```
SQL> SELECT NOM_PRODUIT,
  2         CODE_CATEGORIE,
  3         NO_FOURNISSEUR,
  4         TO_CHAR( AVG ( UNITES_STOCK ) OVER
  5                   ( PARTITION BY CODE_CATEGORIE ), '999D00'),
  6         TO_CHAR(AVG(UNITES_STOCK) OVER
  7                   ( PARTITION BY NO_FOURNISSEUR ), '999D00' ),
  8         SUM(UNITES_STOCK) OVER ()
  9  FROM PRODUITS
 10  WHERE INDISPONIBLE = 0;
```

Exercice n° 2 Le calcul cumulatif

Écrivez les requêtes permettant d'afficher :

– Le client, les commandes, l'année, le mois, les frais de port, la somme cumulative des frais de port pour le mois en cours et la somme cumulative des frais de port pour l'année.

```
SQL> SELECT CODE_CLIENT, NO_COMMANDE,
  2     EXTRACT( YEAR FROM DATE_COMMANDE),
  3     EXTRACT( MONTH FROM DATE_COMMANDE),
  4     TO_CHAR( PORT,'99g999D00U') "Frais Port",
  5     TO_CHAR( SUM(PORT)
  6        OVER( PARTITION BY CODE_CLIENT,
  7                           EXTRACT( MONTH FROM DATE_COMMANDE)
  8              ORDER BY     CODE_CLIENT,
  9                           EXTRACT( MONTH FROM DATE_COMMANDE),
 10                           NO_COMMANDE),'99G999D00U') "S Mois",
 11     TO_CHAR( SUM(PORT)
 12        OVER( PARTITION BY CODE_CLIENT,
 13                           EXTRACT( YEAR FROM DATE_COMMANDE)
 14        ORDER BY           CODE_CLIENT,
 15                           EXTRACT( YEAR FROM DATE_COMMANDE),
 16                           EXTRACT( MONTH FROM DATE_COMMANDE),
```

```
 17                            NO_COMMANDE),
 18                            '9999G999D00U') "S Année"
 19  FROM COMMANDES
 20  ORDER BY CODE_CLIENT, NO_COMMANDE, DATE_COMMANDE;
```

– Le fournisseur, le nom du produit, les unités commandées, la somme cumulative des unités commandées pour le même fournisseur et la somme cumulative des unités commandées pour tous les fournisseurs et produits.

```
SQL> SELECT NO_FOURNISSEUR,
  2         NOM_PRODUIT,
  3         TO_CHAR( UNITES_COMMANDEES,'9999'),
  4         TO_CHAR( SUM ( UNITES_COMMANDEES ) OVER
  5             ( PARTITION BY NO_FOURNISSEUR ORDER BY NOM_PRODUIT),
  6              '99G999'),
  7         SUM(UNITES_COMMANDEES) OVER (
  8                              ORDER BY NO_FOURNISSEUR,NOM_PRODUIT)
  9  FROM PRODUITS
 10  WHERE UNITES_COMMANDEES IS NOT NULL
 11  ORDER BY NO_FOURNISSEUR,NOM_PRODUIT;
```

Exercice n° 3 Le fenêtrage

Écrivez les requêtes permettant d'afficher :

– L'année, le mois, la somme des frais de port, la moyenne des frais de port pour les trois derniers mois à partir du mois en cours.

```
SQL> SELECT ANNEE "Année",
  2        MOIS "Mois",
  3        TO_CHAR( SUM(PORT),'99G999D00U'),
  4        TO_CHAR( AVG(SUM(PORT)) OVER (
  5                   ORDER BY ANNEE
  6                   ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)
  7                   ,'99G999D00U')
  8  FROM COMMANDES
  9  GROUP BY ANNEE, MOIS
 10  ORDER BY ANNEE, MOIS;
```

– L'année, le mois, la somme des frais de port, la moyenne des frais de port pour six mois glissants, les trois derniers mois et les trois mois suivants à partir du mois en cours.

```
SQL> SELECT ANNEE "Année",
  2        MOIS "Mois",
  3        TO_CHAR( SUM(PORT),'99G999D00U'),
  4        TO_CHAR( AVG(SUM(PORT)) OVER (
  5                   ORDER BY ANNEE
  6                   ROWS BETWEEN 2 PRECEDING AND 3 FOLLOWING)
  7                   ,'99G999D00U')
  8  FROM COMMANDES
  9  GROUP BY ANNEE, MOIS
 10  ORDER BY ANNEE, MOIS;
```

Atelier 9.2 Les fonctions analytiques

Questions

1. Quelle est la fonction qui affecte à chaque ligne un numéro unique, déterminé par la clause « **ORDER BY** » ?

   Réponse : ROW_NUMBER

2. Quelle est la fonction qui calcule le ratio d'une valeur par rapport à la somme d'un jeu de valeurs ?

   Réponse : RATIO_TO_REPORT

3. Quelle est la fonction qui fournit un accès à une ligne à un espacement donné après la position courante ?

   Réponse : LEAD

4. Quelle est la fonction qui calcule la valeur d'un rang au sein d'un groupe de valeurs qui ne laisse aucun vide dans la séquence de classement. ?

   Réponse : DENSE_RANK

5. Quelle est la fonction qui effectue un partitionnement ordonné en un nombre spécifique de groupes ?

   Réponse : NTILE

Exercice n° 1 Le classement

Écrivez les requêtes permettant d'afficher :

– Le nom du produit, les unités en stock, le classement des volumes de stocks par produits et le pourcentage du poids du stock d'un produit dans le volume total.

```
SQL> SELECT NOM_PRODUIT,
  2         UNITES_STOCK,
  3         DENSE_RANK() OVER (ORDER BY UNITES_STOCK ) "Le rang",
  4         TO_CHAR( PERCENT_RANK() OVER (ORDER BY UNITES_STOCK ),
  5                  '0D000')||'%' "% du Total"
  6  FROM PRODUITS;
```

– L'année, le mois, la somme des frais de port, le classement de sommes des frais de port pour l'ensemble des valeurs ainsi que le classement un pourcentage pour l'ensemble des valeurs.

```
SQL> SELECT ANNEE "Année",
  2         MOIS "Mois",
  3         TO_CHAR( SUM(PORT),'99G999D00U') "Frais de port",
  4         DENSE_RANK() OVER (ORDER BY SUM(PORT))"Le rang",
  5         TO_CHAR( PERCENT_RANK() OVER (ORDER BY SUM(PORT) ),
  6                     '0D000')||'%' "% du Total"
  7  FROM COMMANDES
  8  GROUP BY ANNEE,MOIS;
```

– L'année, le mois, la somme des frais de port, le classement de sommes des frais de port pour l'ensemble des valeurs de l'année et le classement en pourcentage pour l'ensemble des valeurs de l'année.

```
SQL> SELECT EXTRACT( YEAR FROM DATE_COMMANDE) "Année",
  2         EXTRACT( MONTH FROM DATE_COMMANDE)"Mois",
  3         TO_CHAR( SUM(PORT),'99G999D00U'),
  4         DENSE_RANK() OVER (
  5                    PARTITION BY EXTRACT(YEAR FROM DATE_COMMANDE)
  6                    ORDER BY SUM(PORT))  "Le rang",
  7         TO_CHAR( PERCENT_RANK() OVER (
  8                   PARTITION BY EXTRACT(YEAR FROM DATE_COMMANDE)
  9                   ORDER BY SUM(PORT) ),
 10                  '0D000')||'%' "% du Total"
 11  FROM COMMANDES
 12  GROUP BY EXTRACT( YEAR FROM DATE_COMMANDE),
 13           EXTRACT( MONTH FROM DATE_COMMANDE);
```

– Le client, le nombre des commandes, la somme des frais de port, le classement suivant le nombre des commandes pour l'ensemble des valeurs et le classement en dix groupes des clients suivant leur nombre des commandes.

```
SQL> SELECT CODE_CLIENT,
  2         SUM(NO_COMMANDE),
  3         TO_CHAR( SUM(PORT),'99G999D00U'),
  4         DENSE_RANK() OVER ( ORDER BY SUM(PORT))  "Le rang",
  5         NTILE(10) OVER ( ORDER BY SUM(PORT))     "Le groupe"
  6  FROM COMMANDES
  7  GROUP BY CODE_CLIENT;
```

Exercice n° 2 Les fonctions de fenêtre

Écrivez les requêtes permettant d'afficher :

– L'année, le mois, la somme des frais de port, les frais de port du mois précédent.

```
SQL> SELECT EXTRACT ( YEAR FROM DATE_COMMANDE) "Année",
  2         EXTRACT ( MONTH FROM DATE_COMMANDE) "Mois",
  3         SUM(PORT) "Port",
  4         FIRST_VALUE( SUM(PORT)) OVER
  5              ( ORDER BY EXTRACT ( YEAR FROM DATE_COMMANDE),
  6                         EXTRACT ( MONTH FROM DATE_COMMANDE)
  7                ROWS BETWEEN 1 PRECEDING AND UNBOUNDED FOLLOWING)
  8                "Mois précédent"
  9  FROM COMMANDES
 10  GROUP BY EXTRACT ( YEAR FROM DATE_COMMANDE),
 11           EXTRACT ( MONTH FROM DATE_COMMANDE)
 12  ORDER BY EXTRACT ( YEAR FROM DATE_COMMANDE),
 13           EXTRACT ( MONTH FROM DATE_COMMANDE);
```

– L'année, le mois, la somme des frais de port, les frais de port du premier mois de l'année, ainsi que les frais de port du dernier mois de l'année.

```
SQL> SELECT EXTRACT ( YEAR FROM DATE_COMMANDE) "Année",
  2         EXTRACT ( MONTH FROM DATE_COMMANDE) "Mois",
  3         SUM(PORT) "Port",
  4         FIRST_VALUE( SUM(PORT)) OVER
  5          ( PARTITION BY EXTRACT ( YEAR FROM DATE_COMMANDE)
  6            ORDER BY EXTRACT ( MONTH FROM DATE_COMMANDE)
  7            ROWS BETWEEN UNBOUNDED PRECEDING AND
  8            UNBOUNDED FOLLOWING) "Premier mois de l'année",
```

```
 9         LAST_VALUE( SUM(PORT)) OVER
10          ( PARTITION BY EXTRACT ( YEAR FROM DATE_COMMANDE)
11            ORDER BY EXTRACT ( MONTH FROM DATE_COMMANDE)
12            ROWS BETWEEN UNBOUNDED PRECEDING AND
13            UNBOUNDED FOLLOWING)"Dernier mois de l'année"
14  FROM COMMANDES
15  GROUP BY EXTRACT ( YEAR FROM DATE_COMMANDE),
16           EXTRACT ( MONTH FROM DATE_COMMANDE)
17  ORDER BY EXTRACT ( YEAR FROM DATE_COMMANDE),
18           EXTRACT ( MONTH FROM DATE_COMMANDE);
```

– L'année, le mois, la somme des frais de port, la croissance mensuelle (la différence entre les frais de port pour le mois en cours et les frais de port du mois précédent de l'année uniquement, s'il s'agit du premier mois de l'année, on affiche 0).

```
SQL> SELECT EXTRACT ( YEAR FROM DATE_COMMANDE) "Année",
  2         EXTRACT ( MONTH FROM DATE_COMMANDE) "Mois",
  3         SUM(PORT) "Port",
  4         SUM(PORT) -
  5         CASE LAG( SUM(PORT), 1, 0) OVER
  6              ( PARTITION BY EXTRACT ( YEAR FROM DATE_COMMANDE)
  7                ORDER BY EXTRACT ( MONTH FROM DATE_COMMANDE))
  8         WHEN 0 THEN
  9           SUM(PORT)
 10         ELSE
 11           LAG( SUM(PORT), 1, 0) OVER
 12                ( PARTITION BY EXTRACT ( YEAR FROM DATE_COMMANDE)
 13                  ORDER BY EXTRACT ( MONTH FROM DATE_COMMANDE))
 14         END "Croissance mensuelle"
 15  FROM COMMANDES
 16  GROUP BY EXTRACT ( YEAR FROM DATE_COMMANDE),
 17           EXTRACT ( MONTH FROM DATE_COMMANDE)
 18  ORDER BY EXTRACT ( YEAR FROM DATE_COMMANDE),
 19           EXTRACT ( MONTH FROM DATE_COMMANDE);
```

Atelier 10.1

Questions

1. Quelles sont les syntaxes incorrectes ?

   A.
   ```
   SELECT ANNEE, MOIS, PORT FROM COMMANDES
   MODEL DIMENSION BY (MOIS) MEASURES (P) RULES();
   ```

   B.
   ```
   SELECT MOIS, PORT FROM COMMANDES
   MODEL DIMENSION BY (MOIS) MEASURES (P) RULES();
   ```

   C.
   ```
   SELECT ANNEE, MOIS, PORT FROM COMMANDES
   MODEL DIMENSION BY (ANNEE,MOIS) MEASURES (P)
   RULES();
   ```

   D.
   ```
   SELECT ANNEE, MOIS, PORT FROM COMMANDES
   MODEL DIMENSION BY (MOIS) MEASURES (P)
   RULES(ANNEE);
   ```

   Réponse : A, D

2. Quelles sont les syntaxes incorrectes ?

   A.
   ```
   ... RULES UPSERT ALL ( P[ANY ,'Total'] =
                          SUM(P)[ANY,ANY] )
   ```

   B.
   ```
   ... RULES UPSERT ALL ( P[ANY ,'Total'] =
                          P[ANY,ANY] )
   ```

   C.
   ```
   ... RULES ( P[ANY ,'Total'] =
                          SUM(P)[ANY,ANY] )
   ```

   Réponse : B, C

Exercice n° 1

Écrivez les requêtes permettant d'afficher :

– Les frais de port mensuels en 2011 pour les ventes réalisées en France et Autriche. Vous devez initialiser une nouvelle colonne qui affiche la difference entre les frais de port pour les ventes aux clients autrichiens et aux clients français.

```
SQL> WITH PPA_AUT AS (
  2    SELECT PAYS, MOIS, SUM(PORT) P
  3                    FROM COMMANDES NATURAL JOIN CLIENTS
  4                    WHERE PAYS IN ('Autriche','France')
  5                      AND ANNEE = 2011
  6                    GROUP BY PAYS, MOIS)
  7  SELECT * FROM PPA_AUT
  8  MODEL DIMENSION BY (PAYS,MOIS) MEASURES (P, 0 DIFFERENCE)
  9     RULES ( DIFFERENCE[ANY,ANY] =
 10                          P['Autriche',CV()]-P['France',CV()]);
```

– Les quantites vendues aux les clients francais dans le mois de janvier 2010 et afficher la somme pour les catégories '`Viandes`', '`Viande en conserve`' et '`Poissons et fruits de mer`' dans une nouvelle ligne avec le nom de catégorie `'Viandes et Poissons'`.

```
SQL> WITH
  2    CQ_2010_01 AS (
  3    SELECT NOM_CATEGORIE CATEGORIE, SUM(DC.QUANTITE) Q
  4    FROM CLIENTS CL NATURAL JOIN COMMANDES CO
  5            NATURAL JOIN DETAILS_COMMANDES DC
  6                    JOIN PRODUITS   USING(REF_PRODUIT)
  7                    JOIN CATEGORIES USING(CODE_CATEGORIE)
  8    WHERE PAYS = 'France' AND ANNEE = 2010 AND MOIS = 1
  9    GROUP BY NOM_CATEGORIE)
 10    SELECT CATEGORIE, Q FROM CQ_2010_01
 11        MODEL DIMENSION BY (CATEGORIE) MEASURES (Q)
 12        RULES ( Q['Viandes et Poissons'] =
 13                Q['Viandes']+
 14                Q['Viande en conserve']+
 15                Q['Poissons et fruits de mer'])
 16    ORDER BY CATEGORIE;
```

– La somme des frais de port mensuels pour les clients : français, espagnols et italiens pour toutes les années. Affichez une nouvelle colonne avec la valeur des frais de port. Pour le mois de janvier, juillet et décembre augmentez la somme de frais de port affichée dans la colonne initiale. Pour chaque année, ajoutez une ligne de plus, le mois 13, affichant la somme annuelle des deux colonnes de métriques.

```
SQL> WITH
  2    VENTES AS (
  3        SELECT PAYS, ANNEE, MOIS, SUM(PORT) P
  4        FROM CLIENTS CL NATURAL JOIN COMMANDES
  5        WHERE PAYS IN ('France','Espagne','Italie')
  6        GROUP BY PAYS, ANNEE, MOIS)
  7  SELECT * FROM VENTES
  8  MODEL PARTITION BY (PAYS) DIMENSION BY (ANNEE, MOIS)
  9  MEASURES (P,0 O)
 10  RULES UPSERT ALL SEQUENTIAL ORDER
 11     ( O[ANY,ANY]=P[CV(),CV()],
 12       P[2010,FOR MOIS IN ( 1,7,12)]= P[CV(),CV()]*1.1,
 13       P[2011,1]= P[CV(),CV()]*1.1,
 14       O[ANY,13]=SUM(O)[CV(),ANY],
 15       P[ANY,13]=SUM(P)[CV(),ANY])
 16  ORDER BY 1,2,3;
```

Atelier 11.1

Exercice n° 1

Écrivez les requêtes permettant d'afficher :

– Le cumul des ventes par catégorie de produits par pays uniquement pour les ventes du `2011` et les pays `'Allemagne'`, `'Espagne'` et `'France'`. Présentez les noms des catégories en lignes et les pays en colonnes.

```
SQL> WITH VPC AS (SELECT PAYS, NOM_CATEGORIE CATEGORIE,
  2         SUM(DC.PRIX_UNITAIRE*DC.QUANTITE) CA
  3         FROM CLIENTS CL NATURAL JOIN COMMANDES CO
  4              NATURAL JOIN DETAILS_COMMANDES DC
  5                           JOIN PRODUITS   USING(REF_PRODUIT)
  6                           JOIN CATEGORIES USING(CODE_CATEGORIE)
  7          WHERE PAYS IN ('Allemagne','France','Espagne')
  8          AND ANNEE = 2011
  9          GROUP BY PAYS, NOM_CATEGORIE)
 10  SELECT * FROM VPC PIVOT ( SUM(CA) FOR PAYS
 11              IN ( 'Allemagne' as Allemagne,
 12                   'Espagne'   as Espagne,
 13                   'France'    as France))
 14  ORDER BY CATEGORIE;
```

– Les quantités vendues par catégorie de produits par année et par pays, uniquement pour les clients italiens et espagnols. Vous devez afficher le nom des catégories en lignes et les années et le pays en colonnes.

```
SQL> WITH
  2    CQ_2010_01 AS (
  3    SELECT NOM_CATEGORIE CATEGORIE, SUM(DC.QUANTITE) Q
  4    FROM CLIENTS CL NATURAL JOIN COMMANDES CO
  5             NATURAL JOIN DETAILS_COMMANDES DC
  6                      JOIN PRODUITS   USING(REF_PRODUIT)
  7                      JOIN CATEGORIES USING(CODE_CATEGORIE)
  8    WHERE PAYS = 'France' AND ANNEE = 2010 AND MOIS = 1
  9    GROUP BY NOM_CATEGORIE)
 10    SELECT CATEGORIE, Q FROM CQ_2010_01
 11        MODEL DIMENSION BY (CATEGORIE) MEASURES (Q)
 12        RULES ( Q['Viandes et Poissons'] =
 13                Q['Viandes']+
 14                Q['Viande en conserve']+
 15                Q['Poissons et fruits de mer'])
 16    ORDER BY CATEGORIE;
```

– L'arborescence des employés en commencent par `'Giroux'` et formaté de la sorte :

```
N NOM
-- ------------------------------
 1 Giroux
 2 -----Fuller
 3 ----------Splingart
 4 ---------------Cleret
```

```
 4 ---------------Poidatz
 4 ---------------Capharsie
 4 ---------------Chaussende
 4 ---------------Hanriot
 4 ---------------Jenny
 4 ---------------Steiner
 4 ---------------Maurousset
 4 ---------------Montesinos
 4 ---------------Marchand
...
```

Réponse :

```
SQL> SELECT LEVEL N, LPAD(NOM,LENGTH(NOM)+(LEVEL-1)*5,'-') NOM
  2  FROM EMPLOYES
  3  CONNECT BY PRIOR NO_EMPLOYE = REND_COMPTE
  4  START WITH NOM = 'Giroux';
```

Atelier 12.1 La manipulation des données

Questions

1. Sachant que la table COMMANDES vient d'être créée, quelles sont les requêtes valides pour insérer des valeurs dans cette table ?

COMMANDES			
NO_COMMANDE	NUMBER(6)	<pk>	not null
CODE_CLIENT	CHAR(5)		not null
NO_EMPLOYE	NUMBER(2)		not null
DATE_COMMANDE	DATE		not null
DATE_ENVOI	DATE		null
PORT	NUMBER(8,2)		null

A.
```
INSERT INTO COMMANDES VALUES
( 1,'PERIC',8,'14/13/96','20/11/96', 2.69 );
```

B.
```
INSERT INTO COMMANDES VALUES
( 2,'ANTON',3,'27/11/96',NULL, 1.10 );
```

C.
```
INSERT INTO COMMANDES VALUES
( 3,'BOTTM',3,'10/01/97','15/01/97', .123 );
```

D.
```
INSERT INTO COMMANDES VALUES
( 5,'HUNGC',3,,'24/01/97', .01 );
```

E.
```
INSERT INTO COMMANDES VALUES
( 4,'FURIB',4,NULL,'14/03/97', 4.45 );
```

F.
```
INSERT INTO COMMANDES VALUES
( 6,'SIMOB',223,'03/06/97','13/06/97', .49 );
```

G.
```
INSERT INTO COMMANDES VALUES
( 72,NULL,8,'12/03/97','19/03/97', .21 );
```

Réponse : B, C

2. Quels est la valeur insérée dans le champ `PORT` par la requête de l'option C de la question précédente ?

A. `0,123`

B. `0`

C. `0,12`

D. `1,23`

E. `NULL`

F. La requête est incorrecte

Réponse : C

3. Quels est la valeur insérée dans le champ `PORT` par la requête de l'option C de la question précédente ?

A.
```
UPDATE EMPLOYES SET SALAIRE = ( SELECT SALAIRE
FROM EMPLOYES  WHERE NO_EMPLOYE = 3)
WHERE NO_EMPLOYE = 2;
```

B.
```
UPDATE EMPLOYES SET SALAIRE = (SELECT SALAIRE
FROM EMPLOYES ) WHERE NO_EMPLOYE = 2;
```

C.
```
UPDATE EMPLOYES SET SALAIRE = (SELECT AVG(SALAIRE)
 FROM EMPLOYES) WHERE NO_EMPLOYE = 2;
```

D.
```
UPDATE EMPLOYES A SET SALAIRE = (SELECT
AVG(SALAIRE) FROM EMPLOYES B
WHERE A.FONCTION = B.FONCTION)
WHERE NO_EMPLOYE = 2;
```

Réponse : B

Exercice n° 1 La mise à jour des données

Insérez une nouvelle catégorie de produits nommée « Légumes et fruits » tout en respectant les contraintes d'insertion et mise à jour de la table `CATEGORIES`, à savoir que le `CODE_CATEGORIE` doit être unique et que les colonnes `NOM_CATEGORIE` et `DESCRIPTION` doivent être renseignées. Affichez l'enregistrement inséré et validez la transaction.

```
SQL> INSERT INTO FOURNISSEURS VALUES
  2              ( 30,'Kelly','707 Oxford Rd.',
  3                'Ann Arbor','48104','Etats-Unis',
  4                '(313) 555-5735','(313) 555-3349' );
```

Le fournisseur `'Nouvelle-Orléans Cajun Delights'` est racheté par le fournisseur `'Grandma Kelly's Homestead'`.

Créez un nouveau fournisseur qui s'appelle « Kelly » avec les mêmes coordonnées que le fournisseur `'Grandma Kelly's Homestead'`.

```
SQL> INSERT INTO FOURNISSEURS VALUES
  2              ( 30,'Kelly','707 Oxford Rd.',
  3                'Ann Arbor','48104','Etats-Unis',
  4                '(313) 555-5735','(313) 555-3349' );
```

Tous les produits livrés anciennement par les fournisseurs `'Nouvelle-Orléans Cajun Delights'` et `'Grandma Kelly's Homestead'` seront distribués par le nouveau fournisseur.

```
SQL> UPDATE PRODUITS SET NO_FOURNISSEUR = 30
  2  WHERE NO_FOURNISSEUR = 2 OR
  3        NO_FOURNISSEUR = 3;
```

Effacez les deux anciens fournisseurs.

```
SQL> DELETE FOURNISSEURS
  2  WHERE NO_FOURNISSEUR = 2 OR
  3        NO_FOURNISSEUR = 3;
```

Affichez les produits livrés par le nouveau fournisseur et exécutez la commande suivante « **`COMMIT ;`** ». (La gestion des transactions fait l'objet du module suivant)

```
SQL> SELECT * FROM PRODUITS
  2  WHERE NO_FOURNISSEUR = 30;
SQL> COMMIT;
```

Exercice n° 2 Les mise à jour évoluées

Introduction

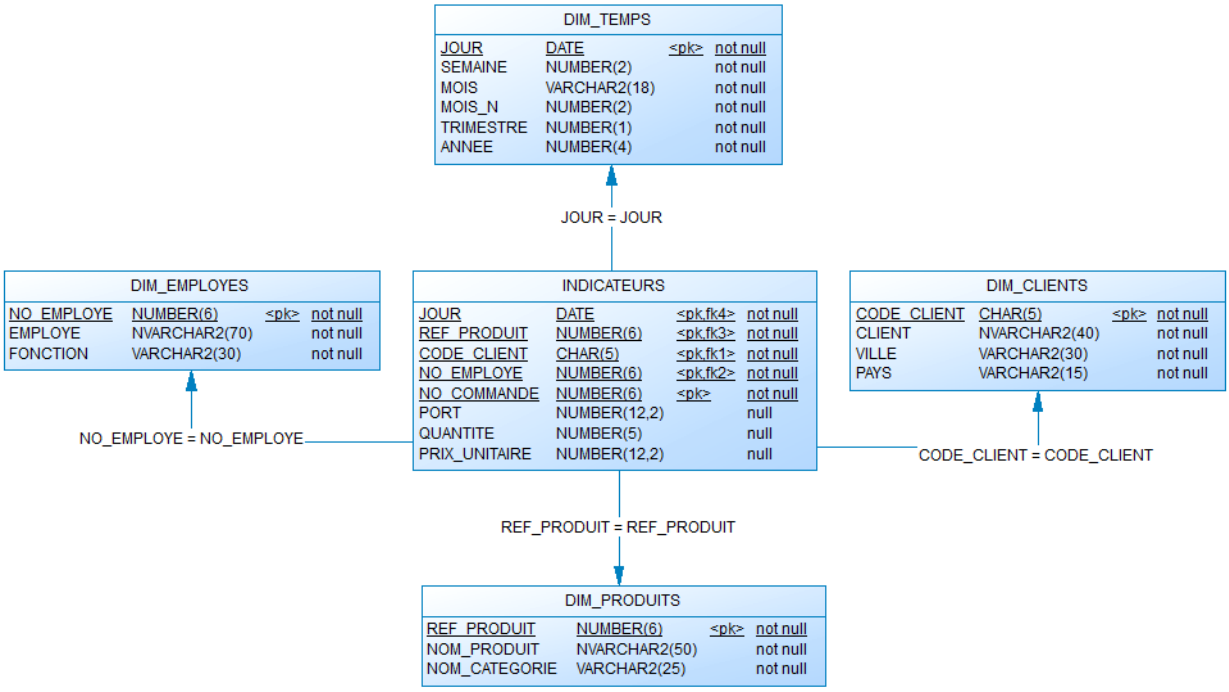
Pour le besoin de l'analyse des données de l'entreprise, un ensemble de tables a été conçu. Elles permettent à l'aide des outils spécialisés d'effectuer une meilleure analyse.

A la création de la base de données, vous avez créé dix-neuf tables. Les sept premières tables ont été utilisées dans les ateliers des modules d'interrogation de données.

Les douze tables restantes sont utilisées dans les ateliers de manipulation des données. Ces tables forment deux ensembles que nous allons alimenter par des scripts.

Pour le bon déroulement des ateliers, il faut d'abord exécuter le script « **`InitEnvEtoileXE.sql`** » qui crée toutes les tables et augmente la taille des tables `COMMANDE` et `DETAILS_COMMANDES`.

La première série des tables est un modèle en étoile simplifiant le modèle logique normalisé en organisant les données de manière optimale pour les traitements d'analyse.



Notez que les champs ont le même nom que ceux du schéma initial sauf les champs suivants :

- `DIM_CLIENTS.CLIENT = CLIENTS.SOCIETE`
- `DIM_EMPLOYES.EMPLOYE = EMPLOYES.NOM||' '||EMPLOYES.PRENOM`

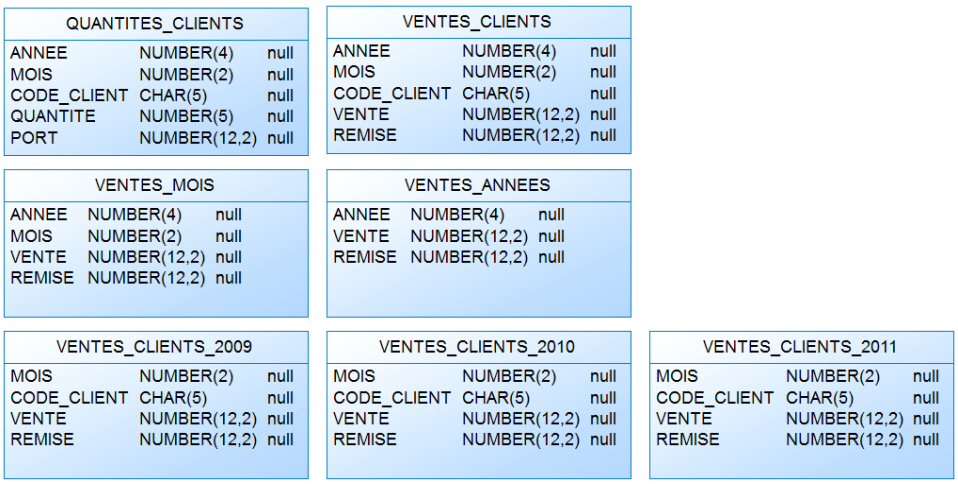
La table `DIM_TEMPS` est déjà alimentée avec les données nécessaires du `01/01/1980'` et jusqu'à `'31/12/2019'`.

La deuxième série des tables est un ensemble des tables récapitulatives pour améliorer les performances des traitements d'analyse. Les données agrégées correspondent à des éléments d'analyse représentatifs des besoins des utilisateurs.

Remarquez que les champs ont le même nom que ceux du schéma initial sauf les champs suivants :

- VENTE = QUANTITE*PRIX_UNITAIRE
- REMISE = QUANTITE*PRIX_UNITAIRE*REMISE

Dans les ateliers suivants, le terme de modèle étoile se réfère à cet ensemble des douze tables, l'ensemble des sept autres tables que vous avez utilisées jusqu'à présent sera nommé le modèle relationnel.



La gestion du modèle étoile

Insérez tous les enregistrements correspondants dans les quatre autres tables restantes DIM_EMPLOYES, DIM_PRODUITS, DIM_CLIENTS et à la fin INDICATEURS.

```
SQL> INSERT INTO DIM_CLIENTS
  2      SELECT CODE_CLIENT, SOCIETE, VILLE, PAYS FROM CLIENTS;

SQL> INSERT INTO DIM_EMPLOYES
  2      SELECT NO_EMPLOYE,NOM||' '||PRENOM,FONCTION FROM EMPLOYES;

SQL> INSERT INTO DIM_PRODUITS
  2      SELECT REF_PRODUIT,NOM_PRODUIT,NOM_CATEGORIE
  3      FROM PRODUITS NATURAL JOIN CATEGORIES;

SQL> INSERT INTO INDICATEURS
  2  SELECT DATE_COMMANDE, REF_PRODUIT, CODE_CLIENT, NO_EMPLOYE,
  3         NO_COMMANDE, PORT, QUANTITE, PRIX_UNITAIRE
  4  FROM EMPLOYES JOIN COMMANDES          USING ( NO_EMPLOYE )
  5                JOIN DETAILS_COMMANDES USING ( NO_COMMANDE );

SQL> COMMIT;
```

Les données de production peuvent être modifiées, les données de ce modèle en étoile peuvent également être utilisées pour des simulations nécessaires à l'analyse. Ainsi les valeurs des enregistrements du modèle en étoile divergent des données de production.

Ecrivez les requêtes permettant de mettre à jour les enregistrements et si tel est le cas d'insérer tous les enregistrements manquants dans les quatre tables DIM_EMPLOYES, DIM_PRODUITS, DIM_CLIENTS et à la fin INDICATEURS. Il s'agit de modifier l'instruction « **INSERT** » par « **MERGE** ».

Validez les modifications effectuées.

```
SQL> MERGE INTO  DIM_CLIENTS CIBLE
```

```
  2    USING ( SELECT CODE_CLIENT,SOCIETE,VILLE,PAYS
  3            FROM CLIENTS ) SOURCE
  4    ON ( CIBLE.CODE_CLIENT = SOURCE.CODE_CLIENT)
  5  WHEN MATCHED THEN
  6      UPDATE SET CIBLE.CLIENT = SOURCE.SOCIETE,
  7                 CIBLE.VILLE  = SOURCE.VILLE,
  8                 CIBLE.PAYS   = SOURCE.PAYS
  9  WHEN NOT MATCHED THEN
 10    INSERT ( CODE_CLIENT,CLIENT,VILLE,PAYS)
 11    VALUES ( SOURCE.CODE_CLIENT,SOURCE.SOCIETE,
 12             SOURCE.VILLE,SOURCE.PAYS);

SQL> MERGE INTO DIM_EMPLOYES CIBLE
  2    USING ( SELECT NO_EMPLOYE,NOM||' '||PRENOM EMPLOYE,FONCTION
  3            FROM EMPLOYES ) SOURCE
  4    ON ( CIBLE.NO_EMPLOYE = SOURCE.NO_EMPLOYE)
  5  WHEN MATCHED THEN
  6      UPDATE SET CIBLE.EMPLOYE  = SOURCE.EMPLOYE,
  7                 CIBLE.FONCTION = SOURCE.FONCTION
  8  WHEN NOT MATCHED THEN
  9    INSERT ( NO_EMPLOYE,EMPLOYE,FONCTION)
 10    VALUES ( SOURCE.NO_EMPLOYE,SOURCE.EMPLOYE,SOURCE.FONCTION);

SQL> MERGE INTO DIM_PRODUITS CIBLE
  2    USING ( SELECT REF_PRODUIT,NOM_PRODUIT,NOM_CATEGORIE
  3            FROM PRODUITS NATURAL JOIN CATEGORIES ) SOURCE
  4    ON ( CIBLE.REF_PRODUIT = SOURCE.REF_PRODUIT)
  5  WHEN MATCHED THEN
  6      UPDATE SET CIBLE.NOM_PRODUIT   = SOURCE.NOM_PRODUIT,
  7                 CIBLE.NOM_CATEGORIE = SOURCE.NOM_CATEGORIE
  8  WHEN NOT MATCHED THEN
  9    INSERT ( REF_PRODUIT,NOM_PRODUIT,NOM_CATEGORIE )
 10    VALUES ( SOURCE.REF_PRODUIT,SOURCE.NOM_PRODUIT,
 11             SOURCE.NOM_CATEGORIE);

SQL> MERGE INTO INDICATEURS CIBLE
  2    USING ( SELECT DATE_COMMANDE, REF_PRODUIT, CODE_CLIENT,
  3                   NO_EMPLOYE, NO_COMMANDE, PORT,
  4                   QUANTITE, PRIX_UNITAIRE
  5              FROM EMPLOYES JOIN COMMANDES USING ( NO_EMPLOYE )
  6                            JOIN DETAILS_COMMANDES
  7                   USING ( NO_COMMANDE ) ) SOURCE
  8    ON ( CIBLE.JOUR        = SOURCE.DATE_COMMANDE AND
  9         CIBLE.REF_PRODUIT = SOURCE.REF_PRODUIT   AND
 10         CIBLE.CODE_CLIENT = SOURCE.CODE_CLIENT   AND
 11         CIBLE.NO_EMPLOYE  = SOURCE.NO_EMPLOYE    AND
 12         CIBLE.NO_COMMANDE = SOURCE.NO_COMMANDE )
 13  WHEN MATCHED THEN
 14      UPDATE SET CIBLE.PORT = SOURCE.PORT,
 15                 CIBLE.QUANTITE = SOURCE.QUANTITE,
 16                 CIBLE.PRIX_UNITAIRE = SOURCE.PRIX_UNITAIRE
 17  WHEN NOT MATCHED THEN
 18    INSERT ( JOUR,REF_PRODUIT,CODE_CLIENT,NO_EMPLOYE,
 19             NO_COMMANDE,PORT,QUANTITE,PRIX_UNITAIRE )
```

```
 20     VALUES ( SOURCE.DATE_COMMANDE,SOURCE.REF_PRODUIT,
 21              SOURCE.CODE_CLIENT,SOURCE.NO_EMPLOYE,
 22              SOURCE.NO_COMMANDE,SOURCE.PORT,
 23              SOURCE.QUANTITE,SOURCE.PRIX_UNITAIRE);

SQL> COMMIT;
```

La gestion des tables récapitulatives

Effacez les enregistrements des tables QUANTITES_CLIENTS et VENTES_CLIENTS

Pour optimiser les accès à la base de données, vous devez insérer les cumuls des frais de port et de quantités vendues par client dans la table QUANTITES_CLIENTS. Les cumuls de ventes ainsi que la remise par client sont insérés dans la table VENTES_CLIENTS, à l'aide d'une instruction « **INSERT** » multi-tables. Validez la transaction.

```
SQL> DELETE QUANTITES_CLIENTS;

SQL> DELETE VENTES_CLIENTS;

SQL> INSERT ALL
  2      INTO QUANTITES_CLIENTS
  3           VALUES ( ANNEE, MOIS , CODE_CLIENT, QUANTITE, PORT)
  4      INTO VENTES_CLIENTS
  5           VALUES ( ANNEE, MOIS , CODE_CLIENT, VENTE, REMISE)
  6  SELECT EXTRACT ( YEAR  FROM DATE_COMMANDE) ANNEE,
  7         EXTRACT ( MONTH FROM DATE_COMMANDE) MOIS,
  8         CODE_CLIENT,
  9         SUM(QUANTITE*PRIX_UNITAIRE) VENTE,
 10         SUM(QUANTITE*PRIX_UNITAIRE*REMISE) REMISE,
 11         SUM(QUANTITE) QUANTITE,
 12         SUM(PORT) PORT
 13  FROM   COMMANDES NATURAL JOIN DETAILS_COMMANDES
 14  GROUP BY EXTRACT ( YEAR  FROM DATE_COMMANDE),
 15           EXTRACT ( MONTH FROM DATE_COMMANDE),
 16           CODE_CLIENT
 17  ORDER BY EXTRACT ( YEAR  FROM DATE_COMMANDE),
 18           EXTRACT ( MONTH FROM DATE_COMMANDE),
 19           CODE_CLIENT;

SQL> COMMIT;
```

Pour optimiser les accès à la base de données, vous devez d'abord effacer les enregistrements puis insérer les enregistrements à l'aide d'une instruction « **INSERT** » multi-tables dans les tables :

- VENTES_ANNEES
- VENTES_MOIS
- VENTES_CLIENTS_2009
- VENTES_CLIENTS_2010
- VENTES_CLIENTS_2011

Validez les modifications effectuées.

```
SQL> INSERT FIRST
```

```
  2    WHEN G = 0 AND ANNEE = 2009 THEN
  3         INTO VENTES_CLIENTS_2009
  4              VALUES ( MOIS, CODE_CLIENT, VENTE, REMISE)
  5    WHEN G = 0 AND ANNEE = 2010 THEN
  6         INTO VENTES_CLIENTS_2010
  7              VALUES ( MOIS, CODE_CLIENT, VENTE, REMISE)
  8    WHEN G = 0 AND ANNEE = 2011 THEN
  9         INTO VENTES_CLIENTS_2011
 10              VALUES ( MOIS, CODE_CLIENT, VENTE, REMISE)
 11    WHEN G = 1 THEN
 12         INTO VENTES_MOIS
 13              VALUES ( ANNEE, MOIS, VENTE, REMISE)
 14    WHEN G = 3 THEN
 15         INTO VENTES_ANNEES
 16              VALUES ( ANNEE, VENTE, REMISE)
 17  SELECT GROUPING_ID( ANNEE, MOIS, CODE_CLIENT ) G,
 18        ANNEE,
 19        MOIS,
 20        CODE_CLIENT,
 21        SUM(QUANTITE*PRIX_UNITAIRE) VENTE,
 22        SUM(QUANTITE*PRIX_UNITAIRE*REMISE) REMISE
 23  FROM   COMMANDES NATURAL JOIN DETAILS_COMMANDES
 24  GROUP BY ROLLUP (ANNEE,MOIS,CODE_CLIENT);

SQL> COMMIT;
```

Atelier 13 Les transactions

Questions

1. L'administrateur de la base de données peut-il voir les données en train d'être modifiées dans une transaction par les utilisateurs de la base ?

   Réponse : Non

2. Peut-on annuler partiellement une transaction ?

   Réponse : Oui

3. Quel est le mode de verrouillage par défaut dans Oracle ?

   A. Enregistrement

   B. Table

   C. Segment

   D. Page des données

   Réponse : A

4. Vous avez ouvert deux sessions avec le même utilisateur. Dans la première session, vous modifiez un enregistrement d'une table. Est-ce que dans la deuxième session, connectée avec le même utilisateur, vous pouvez voir la modification effectuée dans l'autre session ?

   Réponse : Non

5. Quelles sont les commandes SQL qui peuvent être annulées dans une transaction ?

   A. `INSERT`

   B. `ALTER`

   C. `CREATE`

   D. `DROP`

   E. `TRUNCATE`

   F. `DELETE`

   G. `UPDATE`

   Réponse : A, F, G

6. Pour les mêmes choix que la question précédente, quelles sont les commandes SQL qui valident automatiquement une transaction ?

   Réponse : B, C, D, E

7. Quelle doit être la valeur de la colonne `SALARY` après l'exécution du script suivant ?

```
SQL> SELECT FIRST_NAME, LAST_NAME, SALARY
  2  FROM HR.EMPLOYEES
  3  WHERE EMPLOYEE_ID = 200;

FIRST_NAME           LAST_NAME                 SALARY
-------------------- ------------------------- ----------
```

```
Jennifer             Whalen                          4400

SQL> UPDATE HR.EMPLOYEES SET SALARY=6000
  2  WHERE EMPLOYEE_ID = 200;

1 ligne mise à jour.

SQL> DROP TABLE SCOTT.EMP;

Table supprimée.

SQL> ROLLBACK;

Annulation (rollback) effectuée.

SQL> SELECT FIRST_NAME, LAST_NAME, SALARY
  2  FROM HR.EMPLOYEES
  3  WHERE EMPLOYEE_ID = 200;

FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------

Jennifer             Whalen                             ?
```

Réponse : 6 000

8. Quelle doit être la valeur de la colonne SALARY après l'exécution du script suivant ?

```
SQL> SELECT FIRST_NAME, LAST_NAME, SALARY
  2  FROM HR.EMPLOYEES
  3  WHERE EMPLOYEE_ID = 200;

FIRST_NAME           LAST_NAME                     SALARY
-------------------- ------------------------- ----------
Jennifer             Whalen                          6000

SQL> UPDATE HR.EMPLOYEES SET SALARY=8000
  2  WHERE EMPLOYEE_ID = 200;

1 ligne mise à jour.

SQL> TRUNCATE TABLE SCOTT.EMP;
TRUNCATE TABLE SCOTT.EMP
                     *
ERREUR à la ligne 1 :
ORA-00942: Table ou vue inexistante

SQL> ROLLBACK;

Annulation (rollback) effectuée.

SQL> SELECT FIRST_NAME, LAST_NAME, SALARY
  2  FROM HR.EMPLOYEES
  3  WHERE EMPLOYEE_ID = 200;
```

```
FIRST_NAME           LAST_NAME                      SALARY
-------------------- ------------------------- ----------

Jennifer             Whalen                            ?
```

Réponse : 8 000

9. Quelle doit être la valeur de la colonne SALARY après l'exécution du script suivant ?

```
SQL> UPDATE HR.EMPLOYEES SET SALARY=5000
  2  WHERE EMPLOYEE_ID = 200;

1 ligne mise à jour.

SQL> SAVEPOINT SP1;

Savepoint créé.

SQL> UPDATE HR.EMPLOYEES SET SALARY=6000
  2  WHERE EMPLOYEE_ID = 200;

1 ligne mise à jour.

SQL> ROLLBACK TO SAVEPOINT SP1;

Annulation (rollback) effectuée.

SQL> SELECT FIRST_NAME, LAST_NAME, SALARY
  2  FROM HR.EMPLOYEES
  3  WHERE EMPLOYEE_ID = 200;

FIRST_NAME           LAST_NAME                      SALARY
-------------------- ------------------------- ----------

Jennifer             Whalen                            ?
```

Réponse : 5 000

Exercice n° 1 Les transactions

Effacez les commandes effectuées par l'employé numéro trois.

L'opération s'est-elle déroulée correctement ? Justifiez votre réponse.

```
SQL> DELETE COMMANDES
  2  WHERE NO_EMPLOYE = 3;
DELETE COMMANDES
*
ERREUR à la ligne 1 :
ORA-02292: violation de contrainte
(STAGIAIRE.FK_DETAILS_COMMANDES_COMMANDES) d'intégrité -
enregistrement fils existant
```

Réponse : L'opération ne peut pas être exécutée, il y a des enregistrements dans la table fils DETAILS_COMMANDES.

Créez deux nouvelles catégories de produits, une `'Boissons non alcoolisées'` et une autre `'Boissons alcoolisées'`; après la création, insérez un point de sauvegarde `POINT_REPERE_1`.

```
SQL> INSERT INTO CATEGORIES VALUES
  2         ( 11,'Boissons non alcoolisées',
  3              'Boissons non alcoolisées' );

1 ligne créée.

SQL> INSERT INTO CATEGORIES VALUES
  2         ( 12,'Boissons alcoolisées',
  3              'Boissons alcoolisées' );

1 ligne créée.

SQL> SAVEPOINT POINT_REPERE_1;

Savepoint créé.
```

Attribuez les produits `1` et `43` à la premiere catégorie et insérez un point de sauvegarde `POINT_REPERE_2`.

```
SQL> UPDATE PRODUITS SET CODE_CATEGORIE = 11
  2  WHERE REF_PRODUIT IN (1,43);

2 ligne(s) mise(s) à jour.

SQL> SAVEPOINT POINT_REPERE_2;

Savepoint créé.
```

Attribuez les produits (`2, 24, 34, 35, 38, 39, 67`) à la deuxième catégorie et insérez un point de sauvegarde `POINT_REPERE_3`.

```
SQL> UPDATE PRODUITS SET CODE_CATEGORIE = 12
  2  WHERE REF_PRODUIT IN ( 2, 24, 34, 35, 38, 39, 67 );

7 ligne(s) mise(s) à jour.

SQL> SAVEPOINT POINT_REPERE_3;

Savepoint créé.
```

Supprimez la catégorie de produits `'Boissons'`.

```
SQL> DELETE CATEGORIES
  2  WHERE CODE_CATEGORIE = 1;
DELETE CATEGORIES
*
ERREUR à la ligne 1 :
ORA-02292: violation de contrainte (STAGIAIRE.FK_PRODUITS_CATEGORIE)
d'intégrité - enregistrement fils existant
```

L'opération s'est déroulée correctement ?

Réponse : Non, il y a encore des produits de la catégorie 1.

Annulez les opérations depuis le point de sauvegarde `POINT_REPERE_2`.

```
SQL> ROLLBACK TO POINT_REPERE_2 ;
```

```
Annulation (rollback) effectuée.
```

Exécutez la commande « **ROLLBACK TO SAVEPOINT POINT_REPERE_3 ;** » Justifiez le message d'erreur.

Réponse : Le POINT_REPERE_3 est ultérieur au POINT_REPERE_2 et il n'est plus reconnu par le système.

Attribuez tous les produits qui sont encore de catégorie « Boissons » à la deuxième catégorie, « Boissons alcoolisées » ; insérez un point de sauvegarde POINT_REPERE_3.

```
SQL> UPDATE PRODUITS SET CODE_CATEGORIE = 12
  2  WHERE CODE_CATEGORIE = 1;

10 ligne(s) mise(s) à jour.
```

Supprimez la catégorie de produits « Boissons ».

```
SQL> DELETE CATEGORIES
  2  WHERE CODE_CATEGORIE = 1;

1 ligne supprimée.
```

Affichez les produits ainsi que les deux catégories qui sont l'objet de cette transaction.

```
SQL> SELECT * FROM CATEGORIES
  2  WHERE CODE_CATEGORIE IN (11,12);

CODE_CATEGORIE NOM_CATEGORIE             DESCRIPTION
-------------- ------------------------- -------------------------
            11 Boissons non alcoolisées  Boissons non alcoolisées
            12 Boissons alcoolisées      Boissons alcoolisées
```

Validez la transaction.

```
SQL> COMMIT;

Validation effectuée.
```

Affichez les enregistrements actuels de la table CATEGORIES. Affichez les enregistrements de la table CATEGORIES telles qu'elles étaient une heure auparavant.

```
SQL> SELECT * FROM CATEGORIES;
SQL> SELECT * FROM CATEGORIES AS OF TIMESTAMP (SYSTIMESTAMP - 1/24);
```

Atelier 14.1 La création des tables

Questions

1. Quels sont les noms de table valides ?

   A. `TEST_DE_NOM_DE_TABLE`

   B. `P#_$TEST_TABLE`

   C. `7_NOM_TABLE`

   D. `SELECT`

   Réponse : A, B

2. Quelles sont les erreurs de syntaxe ou de nom dans la requête suivante ?

```
CREATE TABLE NOUVELLE_TABLE (
         ID NUMBER,
         CHAMP_1 char(40),
         CHAMP_2 char(80),
         ID char(40);
```

   Réponse : Le nom de la colonne ID est dupliqué et il manque une parenthèse avant le point-virgule final.

3. Quelles sont les instructions d'insertion non valides dans la table suivante ?

```
SQL> DESC UTILISATEURS
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- ---------------
 NO_UTILISATEUR                            NOT NULL NUMBER(6)
 NOM_PRENOM                                NOT NULL VARCHAR2(20)
 DATE_CREATION                             NOT NULL DATE
 UTILISATEUR                               NOT NULL VARCHAR2(20)
```

   A.

```
SQL> INSERT INTO UTILISATEURS( NO_UTILISATEUR, NOM_PRENOM)
  2                  VALUES ( 1, 'Razvan BIZOÏ');
```

   B.

```
 SQL> INSERT INTO UTILISATEURS( NO_UTILISATEUR, NOM_PRENOM,
  2          UTILISATEUR) VALUES ( 2, 'Razvan BIZOÏ', 'razvan');
```

   C.

```
 SQL> INSERT INTO UTILISATEURS( NO_UTILISATEUR, NOM_PRENOM,
  2                            DATE_CREATION, UTILISATEUR)
  3  VALUES ( 3, 'Razvan BIZOÏ', 'razvan');
```

   D.

```
 SQL> INSERT INTO UTILISATEURS( NO_UTILISATEUR, DATE_CREATION,
  2               UTILISATEUR)   VALUES ( 4, SYSDATE, 'razvan');
```

   E.

```
SQL> INSERT INTO UTILISATEURS( NO_UTILISATEUR, NOM_PRENOM,
  2  UTILISATEUR)VALUES ( 5, 'BERNHARD Marie-Thérèse', 'razvan');
```

   F.

```
SQL> INSERT INTO UTILISATEURS
  2  VALUES ( 5, 'BERNHARD Marie-Thérèse', 'razvan', sysdate);
```

Réponse : F

4. La syntaxe de création de table suivante est-elle valide ?

```
SQL> CREATE TABLE "Employés"(
  2  "N° employé" NUMBER(6)    NOT NULL,
  3  "Nom"        VARCHAR2(20) NOT NULL,
  4  "Prénom"     VARCHAR2(20) NOT NULL);
```

Réponse : Oui

5. Quelle est la syntaxe correcte pour visualiser les enregistrements de l'exercice précédent ?

A.

```
SQL> SELECT Nom, Prénom FROM Employés;
```

B.

```
SQL> SELECT Nom, Prénom FROM "Employés";
```

C.

```
SQL> SELECT Nom, Prénom FROM Employés;
```

D.

```
SQL> SELECT "Nom", "Prénom"  FROM "Employés";
```

Réponse : D

Exercice n°1 La création des tables

Écrivez les requêtes permettant de créer les tables suivantes :

EX_PRODUITS		
REF_PRODUIT	NUMBER(6)	not null
NOM_PRODUIT	NVARCHAR2(40)	not null
PRIX_UNITAIRE	NUMBER(8,2)	null
UNITES_STOCK	NUMBER(5)	null
DATE_CREATION	DATE	not null

EX_CATEGORIES		
CODE_CATEGORIE	NUMBER(6)	not null
NOM_CATEGORIE	VARCHAR2(25)	not null

Pour la colonne **« DATE_CREATION »** de la table **« EX_PRODUITS »,** initialisez une valeur par défaut égale à la date et l'heure de l'insertion. Les deux tables doivent être stockées dans le tablespace **« GEST_DATA »**.

```
SQL> CREATE TABLE EX_PRODUITS  (
  2     REF_PRODUIT          NUMBER(6)      NOT NULL,
  3     NOM_PRODUIT          NVARCHAR2(40)  NOT NULL,
  4     PRIX_UNITAIRE        NUMBER(8,2),
  5     UNITES_STOCK         NUMBER(5),
  6     DATE_CREATION        DATE           DEFAULT SYSDATE NOT NULL
  7  )  TABLESPACE GEST_DATA;

Table créée.

SQL> CREATE TABLE EX_CATEGORIES  (
  2     CODE_CATEGORIE       NUMBER(6)       NOT NULL,
  3     NOM_CATEGORIE        VARCHAR2(25)    NOT NULL
  4  )  TABLESPACE GEST_DATA;

Table créée.
```

```
SQL> SELECT TABLE_NAME, TABLESPACE_NAME
  2  FROM  USER_TABLES
  3  WHERE TABLE_NAME LIKE 'EX_%';

TABLE_NAME                     TABLESPACE_NAME
------------------------------ ----------------
EX_PRODUITS                    GEST_DATA
EX_CATEGORIES                  GEST_DATA
```

Écrivez la requête qui permet de créer la table **« EX_CLIENTS »** et la table **« EX_FOURNISSEURS »** avec la même structure que les tables **« CLIENTS »** et **« FOURNISSEURS »** sans avoir les enregistrements de ces deux tables. Stockez les deux tables dans le tablespace **« GEST_DATA »**.

```
SQL> CREATE TABLE EX_CLIENTS TABLESPACE GEST_DATA
  2  AS SELECT CODE_CLIENT,SOCIETE,ADRESSE,VILLE,
  3            CODE_POSTAL,PAYS,TELEPHONE,FAX
  4     FROM CLIENTS WHERE 1=2;

Table créée.

SQL> CREATE TABLE EX_FOURNISSEURS TABLESPACE GEST_DATA
  2  AS  SELECT NO_FOURNISSEUR,SOCIETE,ADRESSE,VILLE,
  3             CODE_POSTAL,PAYS,TELEPHONE,FAX
  4      FROM FOURNISSEURS WHERE 1=2;

Table créée.
```

Exercice n°2 Le objets de grande taille

Créez la table **« EX_EMPLOYES »** avec la description suivante :

EX_EMPLOYES		
NO_EMPLOYE	NUMBER(6)	not null
REND_COMPTE	NUMBER(6)	null
NOM	NVARCHAR2(40)	not null
PRENOM	NVARCHAR2(30)	not null
PHOTO	BLOB	null
DESCRIPTION	CLOB	null

Stockez les enregistrements de la colonne **« PHOTO »** dans le tablespace **« GEST_DATA_BLOB »** et les enregistrements de la colonne **« DESCRIPTION »** dans le tablespace **« GEST_DATA_CLOB ».**

```
SQL> CREATE TABLE EX_EMPLOYES (
  2     NO_EMPLOYE           NUMBER(6)           NOT NULL,
  3     REND_COMPTE          NUMBER(6),
  4     NOM                  NVARCHAR2(40)       NOT NULL,
  5     PRENOM               NVARCHAR2(30)       NOT NULL,
  6     PHOTO                BLOB,
  7     DESCRIPTION          CLOB )
  8     TABLESPACE GEST_DATA
  9                STORAGE ( INITIAL 5M )
 10     LOB ( PHOTO)
 11         STORE AS PHOTO
 12         (TABLESPACE GEST_DATA_BLOB
```

```
 13             STORAGE (INITIAL 10M )
 14             CHUNK 4000
 15             NOCACHE NOLOGGING)
 16     LOB ( DESCRIPTION)
 17         STORE AS DESCRIPTION
 18         (TABLESPACE GEST_DATA_CLOB
 19             STORAGE (INITIAL 10M )
 20             CHUNK 4000
 21             NOCACHE NOLOGGING);

Table créée.

SQL> SQL> SELECT TABLE_NAME,SEGMENT_NAME,
  2         TABLESPACE_NAME, INDEX_NAME
  3  FROM USER_LOBS
  4  WHERE TABLE_NAME LIKE 'EX_EMPLOYES';

TABLE_NAME  SEGMENT_NAME TABLESPACE_NAM INDEX_NAME
----------- ------------ ------------- --------------------------
EX_EMPLOYES PHOTO        GEST_DATA_BLOB SYS_IL0000014870C00005$$
EX_EMPLOYES DESCRIPTION  GEST_DATA_CLOB SYS_IL0000014870C00006$$

SQL> SELECT TABLESPACE_NAME,
  2         SEGMENT_NAME, INITIAL_EXTENT
  3  FROM DBA_SEGMENTS
  4  WHERE SEGMENT_NAME IN ( SELECT SEGMENT_NAME
  5                          FROM USER_LOBS
  6                          WHERE TABLE_NAME LIKE 'EX_EMPLOYES') OR
  7        SEGMENT_NAME IN ( SELECT INDEX_NAME
  8                          FROM USER_LOBS
  9                          WHERE TABLE_NAME LIKE 'EX_EMPLOYES') OR
 10        SEGMENT_NAME LIKE 'EX_EMPLOYES';

TABLESPACE_NAM SEGMENT_NAME                 INITIAL_EXTENT
-------------- ---------------------------- --------------
GEST_DATA_CLOB SYS_IL0000014870C00006$$             524288
GEST_DATA_CLOB DESCRIPTION                        10485760
GEST_DATA_BLOB SYS_IL0000014870C00005$$             524288
GEST_DATA      EX_EMPLOYES                         5242880
GEST_DATA_BLOB PHOTO                              10485760
```

Atelier 14.2 La gestion des tables

Questions

Voici différents types de contrainte de la table **« EMPLOYEES »** de l'utilisateur **« HR »**.

```
SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE, DEFERRABLE,
  2         DEFERRED, VALIDATED
  3  FROM DBA_CONSTRAINTS
  4  WHERE OWNER = 'HR' AND TABLE_NAME='EMPLOYEES';

CONSTRAINT_NAME                C DEFERRABLE     DEFERRED  VALIDATED
------------------------------ - -------------- --------- ----------
EMP_LAST_NAME_NN               C NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_EMAIL_NN                   C NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_HIRE_DATE_NN               C NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_JOB_NN                     C NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_SALARY_MIN                 C NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_EMAIL_UK                   U NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_EMP_ID_PK                  P NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_DEPT_FK                    R NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_JOB_FK                     R NOT DEFERRABLE IMMEDIATE VALIDATED
EMP_MANAGER_FK                 R NOT DEFERRABLE IMMEDIATE VALIDATED
```

1. De quel type est la contrainte « **EMP_EMAIL_NN** » ?

   Réponse : De type « **NOT NULL** »

2. Vous avez besoin pour une colonne de vérifier qu'il n'existe pas deux fois la même valeur dans la table en même temps, la colonne ne doit pas contenir des valeurs nulles. Quel est le type de contraintes que vous devez utiliser pour satisfaire les deux conditions ?

   A. CHECK

   B. UNIQUE

   C. NOT NULL

   D. PRIMARY KEY

   E. FOREIGN KEY

   Réponse : D

3. Quel est l'avantage de déclarer une contrainte « **CHECK** » ?

   Réponse : La contrainte « **CHECK** » permet de contrôler la cohérence des données dans une table.

4. Quelle est la différence entre une contrainte « **CHECK** » de colonne et une contrainte « **CHECK** » de table ?

   Réponse : Une contrainte **« CHECK »** de table peut référer plusieurs colonnes.

5. Argumentez pourquoi la syntaxe suivante, de création d'une clé étrangère, est incorrecte ?

```
SQL> CREATE TABLE CATEGORIE (
  2     CODE_CATEGORIE       NUMBER(6)      PRIMARY KEY,
  3     NOM_CATEGORIE        VARCHAR2(25)   NOT NULL);

Table créée.

SQL> CREATE TABLE PRODUIT (
  2     REF_PRODUIT          NUMBER(6)      PRIMARY KEY,
  3     NOM_PRODUIT          VARCHAR2(40)   NOT NULL,
  4     CODE_CATEGORIE       NUMBER(6)      NOT NULL
  5                            CONSTRAINT PRODUITS_CATEGORIES_FK
  6                            FOREIGN KEY
  7                            REFERENCES CATEGORIE);
```

Réponse : Dans le cadre d'une contrainte de type colonne **« FOREIGN KEY »** ne figure pas dans la syntaxe.

6. Quelles sont les requêtes qui créent une table comme la suivante ?

```
SQL> DESC PRODUIT
 Nom                                       NULL ?   Type
 ----------------------------------------- -------- -------------
 REF_PRODUIT                               NOT NULL NUMBER(6)
 NOM_PRODUIT                               NOT NULL VARCHAR2(40)
 CODE_CATEGORIE                            NOT NULL NUMBER(6)
```

A.

```
SQL> CREATE TABLE PRODUIT (
  2     REF_PRODUIT          NUMBER(6)      PRIMARY KEY,
  3     NOM_PRODUIT          VARCHAR2(40)   NOT NULL,
  4     CODE_CATEGORIE       NUMBER(6)      NOT NULL
  5             REFERENCES CATEGORIE ON DELETE SET NULL);
```

B.

```
SQL> CREATE TABLE PRODUIT (
  2     REF_PRODUIT          NUMBER(6)      PRIMARY KEY,
  3     NOM_PRODUIT          VARCHAR2(40)   NOT NULL,
  4     CODE_CATEGORIE       NUMBER(6)
  5             REFERENCES CATEGORIE ON DELETE SET NULL);
```

C.

```
SQL> CREATE TABLE PRODUIT (
  2     REF_PRODUIT          NUMBER(6)      NOT NULL,
  3     NOM_PRODUIT          VARCHAR2(40)   NOT NULL,
  4     CODE_CATEGORIE       NUMBER(6)      NOT NULL
  5             REFERENCES CATEGORIE ON DELETE SET NULL);
```

Réponse : A, C

7. La commande **« DROP TABLE TABLE_NAME »** est-elle équivalente à la commande **« DELETE FROM TABLE_NAME »** ?

Réponse : Non, **« DROP »** détruit l'objet et **« DELETE »** n'efface que les enregistrements.

8. Les colonnes supprimées sont-elle récupérables ?

Réponse : Les colonnes supprimées ne peuvent pas être récupérées.

9. L'activation de la contrainte de la table maître active-t'elle les contraintes d'intégrité référentielle désactivées avec cette contrainte par la clause **« CASCADE »** ?

Réponse : NON

10. Argumentez pourquoi la syntaxe suivante, de suppression de plusieurs colonnes, est incorrecte ?

```
SQL> ALTER TABLE CLIENTS DROP COLUMNS (TELEPHONE ,FAX );
```

Réponse : Lors de la suppression des plusieurs colonnes, le mot clé **« COLUMN »** ne devrait pas être utilisé dans la commande **« ALTER TABLE »**.

11. Décrivez une instruction SQL qui pourrait entraîner le message d'erreur suivant :

```
ERREUR à la ligne 1 : ORA-00955: Ce nom d'objet existe déjà
```

Réponse : La création d'un objet qui existe déjà, une table, un index, une contrainte etc.

12. Décrivez une instruction SQL qui pourrait entraîner le message d'erreur suivant :

```
ERREUR à la ligne 1 :
ORA-02273: cette clé unique/primaire est référencée par des clés
étrangères
```

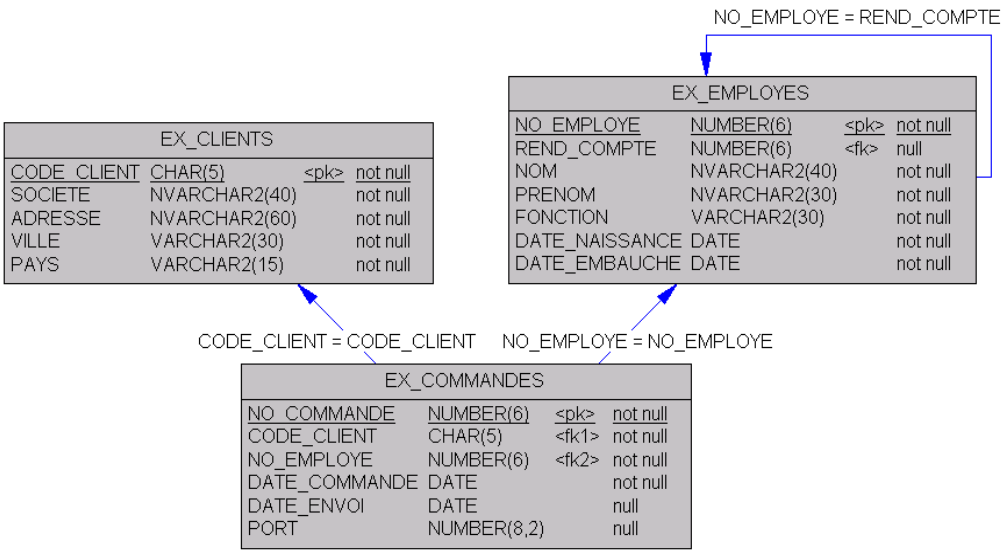
Réponse : Lors de la suppression d'une contrainte de clé primaire, il faut utiliser la clause **« CASCADE »**.

Exercice n° 1 Les contraintes

Effacez les tables **« EX_EMPLOYES »** et **« EX_CLIENTS »** précédemment créée.

```
SQL> DROP TABLE EX_EMPLOYES;
SQL> DROP TABLE EX_CLIENTS;
```

Écrivez les requêtes permettant de créer les tables avec les contraintes suivantes :



Pour la table **« EX_EMPLOYES »**, créez une contrainte **« CHECK »** qui contrôle que l'employé est âgé de dix-huit ans à la **« DATE_EMBAUCHE »**. Pour toutes les contraintes de type **« PRIMARY KEY »,** précisez les informations de stockage de sorte que les index ainsi créés soient stockés dans le tablespace **« GEST_INX »**.

Vous devez stocker les tables **« EX_CLIENTS »** et **« EX_COMMANDES »** dans le tablespace **« GEST_DATA »** et la table **« EX_EMPLOYES »** dans le tablespace **« GEST_ETOILE_DATA ».**

```
SQL> SQL> DROP TABLE EX_COMMANDES PURGE;

Table supprimée.

SQL> DROP TABLE EX_CLIENTS PURGE;

Table supprimée.

SQL> DROP TABLE EX_EMPLOYES PURGE;

Table supprimée.

SQL> CREATE TABLE EX_CLIENTS  (
  2     CODE_CLIENT          CHAR(5)               NOT NULL
  3             CONSTRAINT PK_EX_CLIENTS PRIMARY KEY
  4             USING INDEX  TABLESPACE GEST_INDX,
  5     SOCIETE              NVARCHAR2(40)         NOT NULL,
  6     ADRESSE              NVARCHAR2(60)         NOT NULL,
  7     VILLE                VARCHAR2(30)          NOT NULL,
  8     PAYS                 VARCHAR2(15)          NOT NULL
  9  )  TABLESPACE GEST_DATA;

Table créée.

SQL> CREATE TABLE EX_EMPLOYES  (
  2     NO_EMPLOYE           NUMBER(6)             NOT NULL
  3             CONSTRAINT PK_EX_EMPLOYES PRIMARY KEY
  4             USING INDEX  TABLESPACE GEST_INDX,
  5     REND_COMPTE          NUMBER(6),
  6     NOM                  NVARCHAR2(40)         NOT NULL,
  7     PRENOM               NVARCHAR2(30)         NOT NULL,
  8     FONCTION             VARCHAR2(30)          NOT NULL,
  9     DATE_NAISSANCE       DATE                  NOT NULL,
 10     DATE_EMBAUCHE        DATE  DEFAULT SYSDATE NOT NULL,
 11    CONSTRAINT FK_EX_EMPLOYES_EMPLOYES
 12    FOREIGN KEY (REND_COMPTE)REFERENCES EX_EMPLOYES (NO_EMPLOYE),
 13    CONSTRAINT CHK_EX_EMPLOYES_EMBAUCHE CHECK
 14               (((DATE_EMBAUCHE - DATE_NAISSANCE)  / 365 )  > 18)
 15  ) TABLESPACE GEST_ETOILE_DATA;

Table créée.

SQL> CREATE TABLE EX_COMMANDES  (
  2     NO_COMMANDE          NUMBER(6)             NOT NULL
  3             CONSTRAINT PK_EX_COMMANDES PRIMARY KEY
  4             USING INDEX  TABLESPACE GEST_INDX,
  5     CODE_CLIENT          CHAR(5)               NOT NULL,
  6     NO_EMPLOYE           NUMBER(6)             NOT NULL,
  7     DATE_COMMANDE        DATE                  NOT NULL,
```

```
  8     DATE_ENVOI           DATE,
  9     PORT                 NUMBER(8,2),
 10    CONSTRAINT FK_EX_COMMANDE_CLIENTS
 11    FOREIGN KEY (CODE_CLIENT)REFERENCES EX_CLIENTS (CODE_CLIENT),
 12    CONSTRAINT FK_EX_COMMANDE_EMPLOYES
 13    FOREIGN KEY (NO_EMPLOYE) REFERENCES EX_EMPLOYES (NO_EMPLOYE)
 14  )  TABLESPACE GEST_DATA;

Table créée.

SQL> SELECT TABLE_NAME, TABLESPACE_NAME FROM USER_TABLES
  2  WHERE TABLE_NAME IN
  3           ('EX_CLIENTS','EX_EMPLOYES','EX_COMMANDES');

SQL> SELECT CONSTRAINT_NAME, CONSTRAINT_TYPE,R_CONSTRAINT_NAME,
  2         SEARCH_CONDITION, TABLE_NAME
  3  FROM USER_CONSTRAINTS
  4  WHERE TABLE_NAME IN
  5             ( 'EX_CLIENTS', 'EX_EMPLOYES', 'EX_COMMANDES')
  6  ORDER BY TABLE_NAME DESC;
```

Exercice n°2 La modification d'une table

Déplacez les tables EX_COMMANDES et EX_CLIENTS dans le tablespace GEST_ETOILE_DATA.

```
SQL> ALTER TABLE EX_CLIENTS MOVE TABLESPACE GEST_ETOILE_DATA;

Table modifiée.

SQL> ALTER TABLE EX_COMMANDES MOVE TABLESPACE GEST_ETOILE_DATA;

Table modifiée.
```

Renommez la table EX_EMPLOYES en EX_PERSONNES.

```
SQL> ALTER TABLE EX_EMPLOYES RENAME TO EX_PERSONNES;

Table modifiée.
```

Ajoutez deux colonnes SALAIRE et COMMISSION, à la table EX_PERSONNES, avec la même description que les colonnes de la table EMPLOYES.

```
SQL> ALTER TABLE EX_PERSONNES
  2  ADD ( SALAIRE     NUMBER(8,2) NOT NULL,
  3        COMMISSION  NUMBER(8,2) );

Table modifiée.
```

Supprimez la colonne PAYS, de la table EX_ CLIENTS.

```
SQL> ALTER TABLE EX_CLIENTS DROP COLUMN PAYS;

Table modifiée.
```

Exercice n°3 La modification d'une contrainte

Pour pouvoir insérer les enregistrements dans la table EX_PERSONNES sans tenir compte de l'ordre d'insertion, vous devez invalider la contrainte de clé étrangère sur elle-même (REND_COMPTE = NO_EMPLOYE). Ars l'insertion des enregistrements de la table EMPLOYES, vous devez la réactiver.

```
SQL> ALTER TABLE EX_PERSONNES DISABLE CONSTRAINT
  2        FK_EX_EMPLOYES_EMPLOYES;

SQL> INSERT INTO EX_PERSONNES
  2  SELECT NO_EMPLOYE,REND_COMPTE,NOM,PRENOM,FONCTION,
  3         DATE_NAISSANCE,DATE_EMBAUCHE
  4  FROM EMPLOYES;

SQL> ALTER TABLE EX_PERSONNES ENABLE CONSTRAINT
  2        FK_EX_EMPLOYES_EMPLOYES;
```

Ajoutez une contrainte de type **« CHECK »** qui contrôle l'antériorité de la **« DATE_COMMANDE »** à la **« DATE_ENVOIS »**.

```
SQL> ALTER TABLE EX_COMMANDES ADD CONSTRAINT
  2      CHK_EX_COMMANDES_DATE_COMMANDE CHECK
  3      ( DATE_COMMANDE > DATE_ENVOI ) ENABLE;

Table modifiée.
```

Atelier 15.1 Les vues et autres objets

Questions

1. Décrivez une instruction SQL qui pourrait entraîner le message d'erreur suivant :

```
ERREUR à la ligne 1 :
ORA-01733: les colonnes virtuelles ne sont pas autorisées ici
```

Réponse : La modification des vues en lecture seule.

2. Décrivez une instruction SQL qui pourrait entraîner le message d'erreur suivant :

```
ERREUR à la ligne 1 :
ORA-01402: vue WITH CHECK OPTION - violation de clause WHERE
```

Réponse : La modification d'une vue sans le respect de la clause « **CHECK OPTION** ».

Exercice n°1 La création des vues

Créez une vue de la table des employés affichant les nom et prénom de l'employé ainsi que le nom du supérieur hiérarchique pour les employés de moins de quarante ans.

```
SQL> CREATE OR REPLACE VIEW V_EMPLOYES AS
  2         SELECT A.NOM, A.PRENOM, B.NOM MGR
  3         FROM EMPLOYES A, EMPLOYES B
  4         WHERE A.REND_COMPTE = B.NO_EMPLOYE AND
  5               SYSDATE -  A.DATE_NAISSANCE < 40;

Vue créée.
```

Créez une vue qui permette de valider, en saisie et en mise à jour, des commandes uniquement de l'employé King.

```
SQL> CREATE OR REPLACE VIEW V_COMMANDES AS
  2      SELECT NO_COMMANDE, CODE_CLIENT, NO_EMPLOYE,
  3             DATE_COMMANDE, DATE_ENVOI, PORT
  4      FROM  COMMANDES
  5      WHERE NO_EMPLOYE IN ( SELECT NO_EMPLOYE  FROM EMPLOYES
  6                            WHERE  NOM = 'King')
  7      WITH CHECK OPTION;

Vue créée.
```

Créez une vue qui affiche le nom de la société, l'adresse, le téléphone et la ville des clients qui habitent à Toulouse, à Strasbourg, à Nantes ou à Marseille.

```
SQL> CREATE OR REPLACE VIEW V_CLIENTS AS
  2   SELECT SOCIETE, ADRESSE, TELEPHONE, VILLE
  3   FROM CLIENTS
  4   WHERE VILLE IN ('Toulouse','Strasbourg','Nantes','Marseille');

Vue créée.
```

Créez une vue en lecture seule qui affiche l'ensemble des informations des tables :

- DIM_CLIENTS
- DIM_EMPLOYES
- DIM_TEMPS
- DIM_PRODUITS
- INDICATEURS

Une telle vue est très utile pour une analyse des données de ces tables.

```
SQL> CREATE OR REPLACE
  2    VIEW MODEL_ETOILE( CLIENT,VILLE,PAYS,
  3                       EMPLOYE,FONCTION,
  4                       NOM_PRODUIT, NOM_CATEGORIE,
  5                       NO_COMMANDE,PORT,QUANTITE,PRIX_UNITAIRE,
  6                       JOUR,SEMAINE,MOIS,MOIS_N,TRIMESTRE,ANNEE  )
  7      AS
  8          SELECT CLIENT,VILLE,PAYS,
  9                 EMPLOYE,FONCTION,
 10                 NOM_PRODUIT, NOM_CATEGORIE,
 11                 NO_COMMANDE,PORT,QUANTITE,PRIX_UNITAIRE,
 12                 JOUR,SEMAINE,MOIS,MOIS_N,TRIMESTRE,ANNEE
 13          FROM INDICATEURS JOIN DIM_EMPLOYES USING ( NO_EMPLOYE)
 14               JOIN DIM_CLIENTS  USING( CODE_CLIENT)
 15               JOIN DIM_PRODUITS USING( REF_PRODUIT)
 16               JOIN DIM_TEMPS USING( JOUR)
 17  WITH READ ONLY;

Vue créée.

SQL> DESC MODEL_ETOILE
 Nom                             NULL ?   Type
 ------------------------------- -------- ----------------------
 CLIENT                          NOT NULL NVARCHAR2(40)
 VILLE                           NOT NULL VARCHAR2(30)
 PAYS                            NOT NULL VARCHAR2(15)
 EMPLOYE                         NOT NULL NVARCHAR2(70)
 FONCTION                        NOT NULL VARCHAR2(30)
 NOM_PRODUIT                     NOT NULL NVARCHAR2(50)
 NOM_CATEGORIE                   NOT NULL VARCHAR2(25)
 NO_COMMANDE                     NOT NULL NUMBER(6)
 PORT                                     NUMBER(12,2)
 QUANTITE                                 NUMBER(5)
 PRIX_UNITAIRE                            NUMBER(12,2)
 JOUR                            NOT NULL DATE
 SEMAINE                         NOT NULL NUMBER(2)
 MOIS                            NOT NULL VARCHAR2(18)
 MOIS_N                          NOT NULL NUMBER(2)
 TRIMESTRE                       NOT NULL NUMBER(1)
 ANNEE                           NOT NULL NUMBER(4)
```

Exercice n°2 Les séquences et les synonymes

Créez une séquence pour toutes les clés primaires des tables suivantes :

- EX_COMMANDES
- EX_PERSONNES
- EMPLOYES
- PRODUITS
- FOURNISSEURS

```
SQL> CREATE SEQUENCE SQ_PK_EX_COMMANDES START WITH 1;

Séquence créée.

SQL> CREATE SEQUENCE SQ_PK_EX_PERSONNES START WITH 1;

Séquence créée.

SQL> CREATE SEQUENCE SQ_PK_EMPLOYES START WITH 1;

Séquence créée.

SQL> CREATE SEQUENCE SQ_PK_PRODUITS START WITH 1;

Séquence créée.

SQL> CREATE SEQUENCE SQ_PK_FOURNISSEURS START WITH 1;

Séquence créée.
```

Créez pour toutes les tables des synonymes publics.

```
SQL> SET FEEDBACK OFF
SQL> SET ECHO OFF
SQL> SET PAGESIZE 0
SQL> SET LINESIZE 1500
SQL> SPOOL create_synonyms.sql
SQL> SELECT 'CREATE PUBLIC SYNONYM '||TABLE_NAME||' FOR '||
  2         TABLE_NAME||';'  FROM USER_TABLES
  3  WHERE DROPPED = 'NO';
CREATE PUBLIC SYNONYM EX_DETAILS_COMMANDES FOR EX_DETAILS_COMMANDES;
CREATE PUBLIC SYNONYM EX_CATEGORIES FOR EX_CATEGORIES;
...
SQL> SPOOL OFF
SQL> @create_synonyms.sql
```

Atelier 16.1 Les utilisateurs

Exercice n°1 La création d'un utilisateur

Créez un utilisateur **« APP_USER »** avec le tablespace par défaut **« GEST_DATA »** et le tablespace temporaire par défaut **« TEMP »**.

Forcez l'utilisateur à redéfinir son mot de passe lors de sa prochaine connexion à la base.

Utilisez le profil précédemment créé **« APP_PROF »**.

Accordez-lui le droit de stocker jusqu'à 10 Mb dans le tablespace **« GEST_DATA »** ainsi que dans le tablespace **« GEST_INDX ».**

```
SQL> CREATE USER APP_USER
  2      IDENTIFIED BY OBSOLETTE_PASSWORD1
  3      DEFAULT TABLESPACE GEST_DATA
  4      QUOTA 10M ON GEST_DATA
  5      TEMPORARY TABLESPACE TEMP
  6      QUOTA 10M ON GEST_INDX
  7      PROFILE APP_PROF
  8      PASSWORD EXPIRE;

Utilisateur créé.
```

Exercice n°2 Le test de connexion

Essayez de vous connecter à la base de données. Pourquoi ne pouvez-vous pas vous connecter ?

Réponse : Une fois créé, le compte ne possède aucun droit, et son propriétaire ne peut même pas se connecter tant que ce privilège n'a pas été accordé.

```
SQL> CONNECT APP_USER/OBSOLETTE_PASSWORD1
ERROR:
ORA-28001: le mot de passe est expiré

Modification de mot de passe pour APP_USER
Nouveau mot de passe :
Ressaisir le nouveau mot de passe :
ERROR:
ORA-01045: l'utilisateur APP_USER n'a pas le privilège CREATE
SESSION ; connexion refusée

Mot de passe non modifié
```

Exercice n°3 L'attribution d'un rôle

Accordez le rôle **« CONNECT »** et **« RESOURCE »** à l'utilisateur précédemment créé.

Créez une table à partir du catalogue de l'utilisateur à l'aide de la syntaxe suivante :

```
CREATE TABLE T AS SELECT * FROM CAT ;
```

Affichez l'emplacement de la table.

```
SQL> GRANT CONNECT, RESOURCE TO APP_USER ;

Autorisation de privilèges (GRANT) acceptée.

SQL> CONNECT APP_USER/PASSWORD_1
Connecté.
SQL> CREATE TABLE T1 AS SELECT * FROM CAT ;

Table créée.

SQL> SELECT TABLE_NAME, TABLESPACE_NAME
  2  FROM USER_TABLES
  3  WHERE TABLE_NAME = 'T1';

TABLE_NAME                     TABLESPACE_NAME
------------------------------ ----------------------
T1                             GEST_DATA
```

Exercice n°4 Le verrouillage de compte

Verrouillez le compte **« APP_USER »** ainsi créé.

Essayez de vous connecter.

Connectez-vous avec un compte **« STAGIAIRE »** et déverrouillez le compte **« APP_USER »**.

```
SQL> ALTER USER APP_USER ACCOUNT LOCK ;


SQL> CONNECT APP_USER/PASSWORD_1
ERROR:
ORA-28000: compte verrouillé

Avertissement : vous n'êtes plus connecté à ORACLE.

SQL> CONNECT STAGIAIRE/PWD
Connecté.

SQL> ALTER USER APP_USER ACCOUNT UNLOCK ;

Utilisateur modifié.

SQL> CONNECT APP_USER/PASSWORD_1
Connecté.
```

Exercice n°5 L'effacement d'un utilisateur

Connectez-vous avec le compte **« APP_USER »**.

Essayez d'effacer le compte **« APP_USER ».**

Connectez-vous avec le compte **« STAGIAIRE »** et effacez le compte **« APP_USER »**.

```
SQL> CONNECT APP_USER/PASSWORD_1
Connecté.

SQL> DROP USER APP_USER CASCADE;
DROP USER APP_USER CASCADE
*
ERREUR à la ligne 1 :
ORA-01940: impossible de supprimer un utilisateur qui est connecté

SQL> CONNECT STAGIAIRE/PWD
Connecté.

SQL> DROP USER APP_USER CASCADE;

Utilisateur supprimé.
```

Atelier 16.2 Les privilèges

Exercice n°1

Affichez l'utilisateur **« APP_USER »** s'il existe dans votre base de données.

Créez l'utilisateur en lui octroyant le privilège **« CREATE SESSION »**. Rappelez vous que la commande **« GRANT »** avec l'option **« IDENTIFIED BY »** crée l'utilisateur s'il n'existe pas.

```
SQL> SELECT USERNAME FROM DBA_USERS
  2  WHERE USERNAME LIKE 'APP_USER';

aucune ligne sélectionnée

SQL> GRANT CREATE SESSION TO APP_USER
  2     IDENTIFIED BY PWD;

Autorisation de privilèges (GRANT) acceptée.

SQL> SELECT USERNAME FROM DBA_USERS
  2  WHERE USERNAME LIKE 'APP_USER';

USERNAME
------------------------------
APP_USER
```

Exercice n°2

Créez trois utilisateurs **« APP1 »**, **« APP2 »** et **« APP3 »** à l'aide de la commande **« GRANT »** et octroyez leur le privilège **« CREATE SESSION »**.

```
SQL> GRANT CREATE SESSION TO APP1 IDENTIFIED BY PWD;

Autorisation de privilèges (GRANT) acceptée.

SQL> GRANT CREATE SESSION TO APP2 IDENTIFIED BY PWD;

Autorisation de privilèges (GRANT) acceptée.

SQL> GRANT CREATE SESSION TO APP3 IDENTIFIED BY PWD;

Autorisation de privilèges (GRANT) acceptée.
```

Octroyez à l'utilisateur **« APP1 »** le privilège **« CREATE TABLESPACE »** avec la clause **« WITH ADMIN OPTION »**.

```
SQL> GRANT CREATE TABLESPACE TO APP1 WITH ADMIN OPTION;

Autorisation de privilèges (GRANT) acceptée.
```

Connectez-vous avec l'utilisateur **« APP1 »** et octroyez à l'utilisateur **« APP2 »** le privilège **« CREATE TABLESPACE »** avec la clause **« WITH ADMIN OPTION »**.

```
SQL> CONNECT APP1/PWD
Connecté.
SQL> GRANT CREATE TABLESPACE TO APP2 WITH ADMIN OPTION;

Autorisation de privilèges (GRANT) acceptée.
```

Connectez-vous avec le compte **« STAGIAIRE »** et supprimez l'utilisateur **« APP1 »**.

```
SQL> CONNECT STAGIAIRE/PWD
Connecté.
SQL> DROP USER APP1;

Utilisateur supprimé.
```

Connectez-vous avec l'utilisateur **« APP2 »** et octroyez à l'utilisateur **« APP3 »** le privilège **« CREATE TABLESPACE »** avec la clause **« WITH ADMIN OPTION »**.

La commande aboutit-elle ?

```
SQL> CONNECT APP2/PWD
Connecté.
SQL> GRANT CREATE TABLESPACE TO APP3 WITH ADMIN OPTION;

Autorisation de privilèges (GRANT) acceptée.
```

Réponse : Bien que l'utilisateur **« APP1 »** soit supprimé l'utilisateur **« APP2 »** peut octroyer le privilège **« CREATE TABLESPACE »** avec la clause **« WITH ADMIN OPTION »** à **« APP3 »**.

Exercice n°3

Octroyez à l'utilisateur **« APP2 »** le privilège **« SELECT »** sur la table **« STAGIAIRE.CATEGORIES »** avec la clause **« WITH GRANT OPTION »**.

```
SQL> CONNECT STAGIAIRE/PWD
Connecté.
SQL> GRANT SELECT ON STAGIAIRE.CATEGORIES TO APP2
  2  WITH GRANT OPTION;

Autorisation de privilèges (GRANT) acceptée.
```

Connectez-vous avec l'utilisateur **« APP2 »** et octroyez à l'utilisateur **« APP3 »** le privilège **« SELECT »** sur la table **« STAGIAIRE.CATEGORIES »**. Connectez-vous avec l'utilisateur **« APP3 »** et affichez les noms des catégories.

```
SQL> CONNECT APP2/PWD
Connecté.
SQL> GRANT SELECT ON STAGIAIRE.CATEGORIES TO APP3
  2  WITH GRANT OPTION;

Autorisation de privilèges (GRANT) acceptée.

SQL> CONNECT APP3/PWD
Connecté.
SQL> SELECT NOM_CATEGORIE FROM STAGIAIRE.CATEGORIES;

NOM_CATEGORIE
```

```
------------------------
Boissons
Condiments
Desserts
Produits laitiers
Pâtes et céréales
Viandes
Produits secs
Poissons et fruits de mer

8 ligne(s) sélectionnée(s).
```

Connectez-vous avec le compte **« STAGIAIRE »** et retirez le privilège « **SELECT** » sur la table **« STAGIAIRE.CATEGORIES »,** à l'utilisateur **« APP2 »**.

```
SQL> CONNECT STAGIAIRE/PWD
Connecté.

SQL> REVOKE SELECT ON STAGIAIRE.CATEGORIES FROM APP2;

Suppression de privilèges (REVOKE) acceptée.
```

Connectez-vous avec l'utilisateur **« APP3 »** et interrogez la table **« STAGIAIRE.CATEGORIES »**.

La commande aboutit-elle ?

```
SQL> CONNECT APP3/PWD
Connecté.
SQL> SELECT NOM_CATEGORIE FROM STAGIAIRE.CATEGORIES;
SELECT NOM_CATEGORIE FROM STAGIAIRE.CATEGORIES
                                    *
ERREUR à la ligne 1 :
ORA-00942: Table ou vue inexistante
```

Lorsque l'utilisateur **« APP2 »** est supprimé, l'utilisateur **« APP3 »** perd automatiquement le droit d'accéder à la table **« STAGIAIRE.CATEGORIES »**, c'est également le cas si à la place de supprimer l'utilisateur **« APP2 »,** on lui révoque le privilège **« SELECT »**.