



Oracle 10g SQL Réponses aux ateliers



Atelier 1 Présentation de l'environnement



Questions

1-1.

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.

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.

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.

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.

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.

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.

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.

Réponse : D

1-9.

Réponse : B

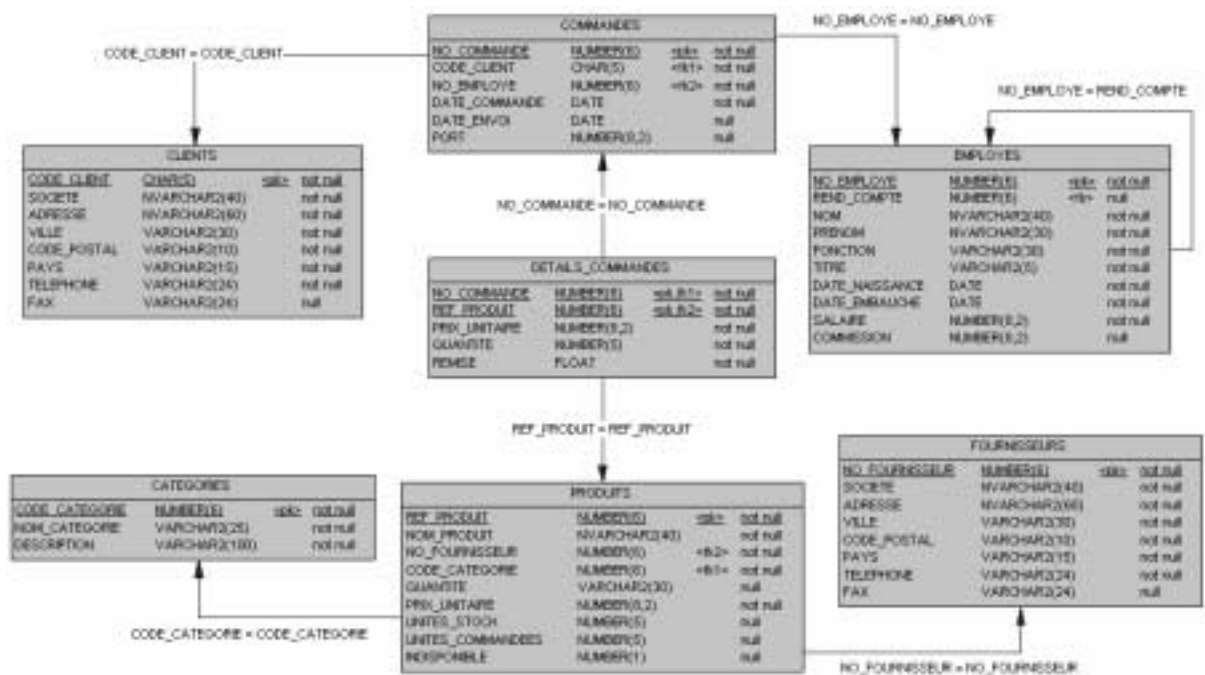
Exercice n° 1 Installation

Installez Oracle XE sur votre machine en tenant compte de votre système d'exploitation. Vous devez suivre attentivement la démarche présentée dans le module.

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

CATEGORIES

FOURNISSEURS

PRODUITS

COMMANDES

CLIENTS

EMPLOYES

EMPLOYES

Enfant

PRODUITS

PRODUITS

DETAILS_COMMANDES

DETAILS_COMMANDES

COMMANDES

COMMANDES

EMPLOYES

Atelier 2 Les outils SQL*Plus



Questions

2-1.

Réponse : A

2-2.

Réponse : Un environnement

2-3.

Réponse : Non

2-4.

Réponse : CONNECT

2-5.

Réponse : OUI

2-6.

Réponse : SPOOL

2-7.

Réponse : OUI

2-8.

Réponse : LINSEIZE, PAGESIZE, FEEDBACK

2-9.

Réponse : DESC

Exercice n° 1 Préparer le poste de développement

Installez le schéma des exemples pour les ateliers en respectant la démarche décrite dans ce module.

```
D:\>dir shema_exemple.zip
Le volume dans le lecteur D s'appelle D0101
Le numéro de série du volume est 0050-009C

Répertoire de D:\

16/07/2006  21:42                73 714 shema_exemple.zip
               1 fichier(s)                73 714 octets
               0 Rép(s)  14 330 048 512 octets libres

D:\>unzip shema_exemple.zip
Archive:  shema_exemple.zip
  creating: Ora10gSQL/
  inflating: Ora10gSQL/db10gStagiaire.sql
  inflating: Ora10gSQL/db10gStagiaireEtoile.sql
  inflating: Ora10gSQL/db10gStagiaireReinitialiseDonnees.sql
```

```

inflating: Ora10gSQL/droptables.sql
inflating: Ora10gSQL/InitEnvStagiaireXE.sql
extracting: Ora10gSQL/InitXE.cmd
inflating: Ora10gSQL/I_CATEGORIES.SQL
inflating: Ora10gSQL/I_CLIENTS.SQL
inflating: Ora10gSQL/I_COMMANDES.SQL
inflating: Ora10gSQL/I_DETAILS_COMMANDES.SQL
inflating: Ora10gSQL/I_EMPLOYES.SQL
inflating: Ora10gSQL/I_FOURNISSEURS.SQL
inflating: Ora10gSQL/I_PRODUITS.SQL

D:\>cd Ora10gSQL

D:\Ora10gSQL>dir
Le volume dans le lecteur D s'appelle D0101
Le numéro de série du volume est 0050-009C

Répertoire de D:\Ora10gSQL

16/07/2006  21:18    <REP>          .
16/07/2006  21:18    <REP>          ..
16/07/2006  21:40         10 529 db10gStagiaire.sql
16/07/2006  21:39          8 850 db10gStagiaireEtoile.sql
16/07/2006  21:27        240 845 db10gStagiaireReinitialiseDonnees.sql
16/07/2006  17:52          2 711 droptables.sql
16/07/2006  21:32         1 151 InitEnvStagiaireXE.sql
16/07/2006  19:03          40 InitXE.cmd
16/04/2003  13:35         683 I_CATEGORIES.SQL
12/07/2006  01:51         13 020 I_CLIENTS.SQL
12/07/2006  01:49         66 784 I_COMMANDES.SQL
16/04/2003  13:55        144 782 I_DETAILS_COMMANDES.SQL
12/07/2006  01:53          1 082 I_EMPLOYES.SQL
16/04/2003  13:59          4 077 I_FOURNISSEURS.SQL
16/04/2003  14:00          7 430 I_PRODUITS.SQL
               13 fichier(s)             501 984 octets
               2 Rép(s)  14 327 549 952 octets libres

D:\Ora10gSQL>sqlplus /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** ».

```

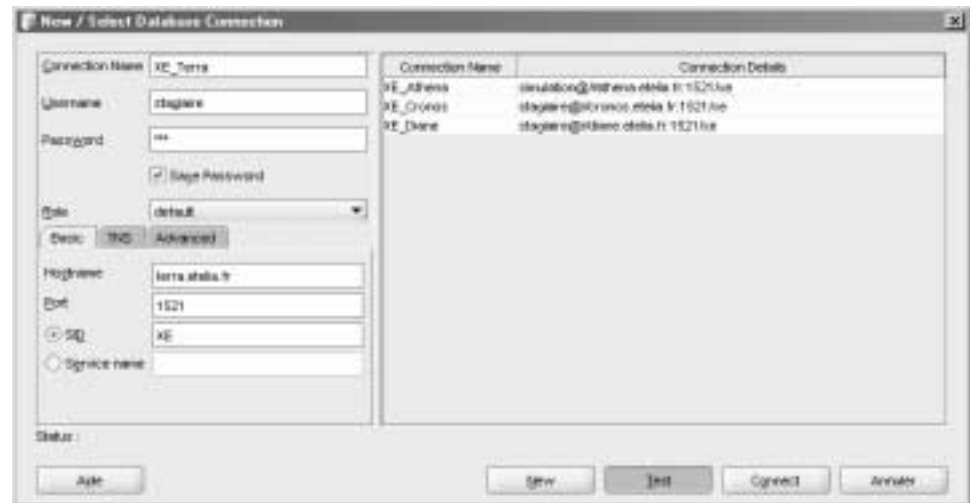
C:\>sqlplus stagiaire/pwd
...
C:\>sqlplus stagiaire/pwd@//diane.etelia.fr:1521/XE
...

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 10g Express Edition Release 10.2.0.1.0
- 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 3.1 Interrogation des données



Questions

3.1-1.

Réponse : Toutes

3.1-2.

Réponse : « ; »

3.1-3.

Réponse : « ' »

3.1-4.

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 3.2 Interrogation des données



Questions

3.2-1.

Réponse : B

3.2-2.

Réponse : NVL

3.2-3.

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 revenus 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      SALAIRE "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 3.3 Interrogation des données



Questions

3.3-1.

Réponse : B, D

3.3-2.

Réponse : A

3.3-3.

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 4.1 Les opérateurs logiques



Questions

4.1-1.

Réponse : Tous

4.1-2.

Réponse : A, B, C, D

4.1-3.

Réponse : E

4.1-4.

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 2.

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

- Le nom, prénom et fonction des employés recrutés après 01/01/1994.

```
SQL> SELECT NOM, PRENOM, FONCTION FROM EMPLOYES
2 WHERE DATE_EMBAUCHE > '01/01/1994';
```

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, 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 '[[[:digit:]]{2}](.[:digit:]]{2}){4}');
```

- Le nom de la société, 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 RECD_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 4.2 Les opérateurs logiques



Questions

4.2-1. ?

Réponse : A, C, D

4.2-2.

Réponse : C

4.2-3.

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 2500 et 3500.

```
SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
2 FROM EMPLOYES
3 WHERE SALAIRE BETWEEN 2500 AND 3500;
```

- Le numéro de commande, code client et la date de commande pour les commandes passées entre le '01/01/1998' et '01/03/1998'.

```
SQL> SELECT NO_COMMANDE, CODE_CLIENT, DATE_COMMANDE
2 FROM COMMANDES
3 WHERE DATE_COMMANDE BETWEEN '01/01/1998' AND '28/02/1998';
```

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 : '18/02/1998', '20/02/1998' ou '25/02/1998'.

```
SQL> SELECT NO_COMMANDE, CODE_CLIENT, DATE_COMMANDE
2 FROM COMMANDES
```

```

3 WHERE DATE_COMMANDE IN
4      ('18/02/1998','20/02/1998','25/02/1998');

```

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 '10/10/1993'.

```

SQL> SQL> SELECT NOM, PRENOM, FONCTION, SALAIRE
2 FROM EMPLOYES
3 WHERE DATE_EMBAUCHE > '10/10/1993' 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 and
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 5 Les chaînes de caractères



Questions

5-1.

Réponse : INITCAP

5-2.

Réponse : A

5-3.

Réponse : A, B

5-4.

Réponse : C

5-5.

Réponse : LTRIM

5-6.

Réponse : REGEXP_REPLACE, TRANSLATE

5-7.

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,'(\(|\)|-|\.|( ))') "Téléphone"
3 FROM FOURNISSEURS;
```

Atelier 6 Les fonctions numériques



Questions

6-1.

Réponse : B, E

6-2.

Réponse : NANVL

6-3.

Réponse : IS NAN, IS INFINITE

6-4.

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 7 Le traitement des dates



Questions

7-1.

Réponse : C

7-2.

Réponse : B

7-3.

Réponse : D

7-4.

Réponse : H

7-5.

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 8.1 Les conversions SQL



Questions

8.1-1.

Réponse : A, E, F, G

8.1-2.

Réponse : A, B, C, E

8.1-3.

Réponse : B

Exercice n° 1 Les conversions

Écrivez les requêtes permettant d'afficher :

- La date du jour formatée de la sorte :

Nous sommes le :

```
-----
Vendredi 14 Juillet 2006
```

```
SQL> SELECT TO_CHAR(SYSDATE,FMDay DD Month YYYY') "Nous sommes le :"  
2 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 8.2 Les conversions SQL



Questions

8.2-1.

Réponse : DECODE, CASE

8.2-2.

Réponse : B

8.2-3.

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 9 Groupement des données



Questions

9-1.

Réponse : C

9-2.

Réponse : A

9-3.

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 SYSTIMESTAMP, DBTIMEZONE, SESSIONTIMEZONE FROM DUAL;
```

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) 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);
```

- 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 cinq produits.

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

- 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 trois fois dans le mois et dont leur frais de port dans le mois sont supérieurs à 1000€

```
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) >= 4 AND
11          SUM(PORT) > 1000;
```

Atelier 10.1 Agrégation et Analyse



Questions

10.1-1.

Réponse : ROLLUP

10.1-2.

Réponse : GROUPING SETS

10.1-3.

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));
```

- 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      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 ROLLUP ( CODE_CLIENT,
8      EXTRACT( YEAR FROM DATE_COMMANDE),
9      EXTRACT( MONTH FROM DATE_COMMANDE));
```

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

```
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 CUBE ( CODE_CLIENT,
```

```

8      EXTRACT( YEAR FROM DATE_COMMANDE),
9      EXTRACT( MONTH FROM DATE_COMMANDE));

```

- 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      EXTRACT( YEAR FROM DATE_COMMANDE) "Année",
4      EXTRACT( MONTH FROM DATE_COMMANDE)"Mois",
5      COUNT(NO_COMMANDE),
6      SUM(PORT)
7  FROM COMMANDES
8  GROUP BY GROUPING SETS (
9      ROLLUP ( NO_EMPLOYE,
10             CODE_CLIENT),
11      ROLLUP ( EXTRACT( YEAR FROM DATE_COMMANDE),
12             EXTRACT( MONTH FROM DATE_COMMANDE)))
13 ORDER BY NO_EMPLOYE,
14          CODE_CLIENT,
15          EXTRACT( YEAR FROM DATE_COMMANDE),
16          EXTRACT( MONTH FROM DATE_COMMANDE);

```

- 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      ALFKI      1998      01      347,65
1      ALFKI      1998      03      202,1
1      ALFKI      1998      ***** 549,75
1      ALFKI      ***** ***** 549,75
1      ANTON      1997      09      20,15
1      ANTON      1997      ***** 20,15
1      ANTON      ***** ***** 20,15
1      AROUT      1997      02      126,8
1      AROUT      1997      06      364,85
1      AROUT      1997      11      118,6
1      AROUT      1997      ***** 610,25
1      AROUT      ***** ***** 610,25
1      BERGS      1997      05      1223,95
1      BERGS      1997      08      693,45
. . .
9      WELLI      1998      02      68,6
9      WELLI      1998      ***** 68,6
9      WELLI      ***** ***** 68,6
9      ***** ***** ***** 16026,41
***** ***** ***** ***** 316266,77

```

```

SQL> SELECT
2      CASE GROUPING( NO_EMPLOYE ) WHEN 0 THEN
3      TO_CHAR(NO_EMPLOYE,'99') 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' );

```

Atelier 10.2 Agrégation et Analyse

Questions



10.2-1.

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

10.2-2.

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 ».

10.2-3.

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> SELECT CODE_CLIENT,
2      NO_COMMANDE,
3      TO_CHAR( PORT,'99g999D00U') "Frais Port",
4      TO_CHAR( AVG(PORT) OVER(PARTITION BY
5          EXTRACT(MONTH FROM DATE_COMMANDE)), '99G999D00U') "Avg Mois",
6      TO_CHAR( AVG(PORT) OVER(PARTITION BY
7          EXTRACT(YEAR FROM DATE_COMMANDE)), '99G999D00U') "Avg Année",
8      COUNT(NO_COMMANDE)OVER(PARTITION BY CODE_CLIENT,
9          EXTRACT(MONTH FROM DATE_COMMANDE)) "Nombre Contrats"
10     FROM COMMANDES
11     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> SELECT NOM,
2      FONCTION,
3      TO_CHAR(SALAIRE,'99G999D00U') "Salaire",
4      TO_CHAR(SALAIRE*100/SUM(SALAIRE) OVER(
5          PARTITION BY FONCTION), '999D00') || '%' "% Fonction",
6      TO_CHAR(SALAIRE*100/SUM(SALAIRE)
7          OVER(), '999D00') || '%' "% Total"
8     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 EXTRACT( YEAR FROM DATE_COMMANDE) "Année",
2          EXTRACT( MONTH FROM DATE_COMMANDE)"Mois",
3          TO_CHAR( SUM(PORT), '99G999D00U'),
4          TO_CHAR( AVG(SUM(PORT)) OVER (
5                      ORDER BY EXTRACT ( YEAR FROM DATE_COMMANDE)
6                      ROWS BETWEEN 3 PRECEDING AND CURRENT ROW)
7                      , '99G999D00U')
8 FROM COMMANDES
9 GROUP BY EXTRACT( YEAR FROM DATE_COMMANDE),
10          EXTRACT( MONTH FROM DATE_COMMANDE)
11 ORDER BY EXTRACT( YEAR FROM DATE_COMMANDE),
12          EXTRACT( MONTH FROM DATE_COMMANDE);
```

- 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 EXTRACT( YEAR FROM DATE_COMMANDE) "Année",
2          EXTRACT( MONTH FROM DATE_COMMANDE)"Mois",
3          TO_CHAR( SUM(PORT), '99G999D00U'),
4          TO_CHAR( AVG(SUM(PORT)) OVER (
5                      ORDER BY EXTRACT ( YEAR FROM DATE_COMMANDE)
6                      ROWS BETWEEN 3 PRECEDING AND 3 FOLLOWING)
7                      , '99G999D00U')
8 FROM COMMANDES
9 GROUP BY EXTRACT( YEAR FROM DATE_COMMANDE),
10          EXTRACT( MONTH FROM DATE_COMMANDE)
11 ORDER BY EXTRACT( YEAR FROM DATE_COMMANDE),
12          EXTRACT( MONTH FROM DATE_COMMANDE);
```

Atelier 10.3 Agrégation et Analyse



Questions

10.3-1.

Réponse : ROW_NUMBER

10.3-2.

Réponse : RATIO_TO_REPORT

10.3-3.

Réponse : LEAD

10.3-4.

Réponse : DENSE_RANK

10.3-5.

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 EXTRACT( YEAR FROM DATE_COMMANDE) "Année",
2         EXTRACT( MONTH FROM DATE_COMMANDE)"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 EXTRACT( YEAR FROM DATE_COMMANDE),
9          EXTRACT( MONTH FROM DATE_COMMANDE);
```

- 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 11.1 Les requêtes multitables

Questions



11.1-1.

Réponse : 1 788 650

11.1-2.

Réponse : B, C, D

11.1-3.

Réponse : A

Exercice n° 1 Les équijointures

Écrivez les requêtes permettant d'afficher :

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

```
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';
```

- La société cliente, le nombre des produits commandés, la ville et le pays qui ont commandé plus de vingt cinq produits.

```
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 AND
5       B.NO_COMMANDE = C.NO_COMMANDE
6 GROUP BY SOCIETE, VILLE, PAYS
7 HAVING COUNT(DISTINCT C.REF_PRODUIT) > 25;
```

- 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.

```
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);
```

- 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).

```
SQL> SELECT A.SOCIETE, B.SOCIETE, A.VILLE
2 FROM FOURNISSEURS A, CLIENTS B
3 WHERE A.VILLE = B.VILLE;
```

- Les sociétés clientes qui ont commandé le produit 'Chai'.

```
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';

```

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.

```

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;

```

- 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.

```

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;

```

- 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.

```

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;

```

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

```

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;

```

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

```

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 (+);

```

Atelier 11.2 Les requêtes multitables



Questions

11.2-1.

Réponse : A, B

11.2-2.

Réponse : E

Exercice n° 1 Les équijointures

Écrivez les requêtes, compatible avec la norme ANSI/ISO SQL : 1999, permettant d'afficher :

- Le nom, le prénom et la société cliente, la date de la commande et les frais de port pour les employés qui ont effectué une vente pour les clients de Paris.

```
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 vingt cinq produits.

```
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 GROUP BY SOCIETE, VILLE, PAYS
5 HAVING COUNT(DISTINCT REF_PRODUIT) > 25;
```

- 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.

```
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).

```
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'.

```
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.

```
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.

```
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.

```
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.

```
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 tel est le cas, ainsi que le nom et prénom des employés gérés par les précédents si tel est le cas.

```
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 12.1 Les jointures complexes



Questions

12.1-1.

Réponse : B

12.1-2.

Réponse : A

12.1-3.

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 12.2 Les jointures complexes



Questions

12.1-1.

Réponse : D, E

12.1-2.

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 '10657'.

```
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 = 10657);
```

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 qui ont commandé le produit 'Chai' mais également qu'ils ont commandé plus de vingt cinq produits.

```
SQL> 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' AND
6 CODE_CLIENT IN (SELECT CODE_CLIENT
```

```

7          FROM CLIENTS JOIN COMMANDES
8              USING (CODE_CLIENT )
9          JOIN DETAILS_COMMANDES
10             USING ( NO_COMMANDE )
11          GROUP BY CODE_CLIENT
12          HAVING COUNT( DISTINCT REF_PRODUIT) > 25);

```

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 TO_CHAR(A.DATE_COMMANDE,'YYYY') =
4      TO_CHAR(B.DATE_COMMANDE,'YYYY') );

```

- 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;
```

ou

```
SQL> SELECT NOM, SALAIRE,  
2         TO_CHAR( 100*SALAIRE/SUM(SALAIRE)  
3                 OVER ( PARTITION BY FONCTION), '999D00')||'%'  
4                 "% total la fonction",  
5         TO_CHAR( 100*SALAIRE/SUM(SALAIRE)  
6                 OVER (), '999D00')||'%' "% total l'entreprise"  
7 FROM EMPLOYES;
```

Atelier 13 Mise à jour des données

Questions



13-1.

Réponse : B, C

13-2.

Réponse : C

13-3.

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

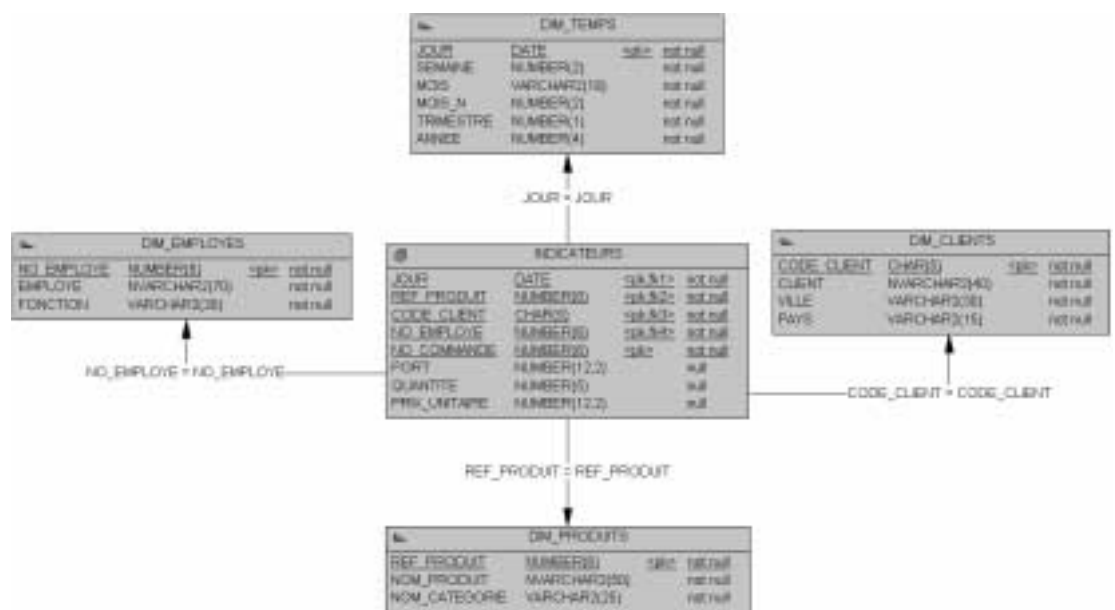
Introduction

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.

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.EMPLOIE = EMPLOYES.NOM || ' ' || EMPLOYES.PRENOM

La table DIM_TEMPS est déjà alimentée avec les données nécessaires du '01 janvier 1996' et jusqu'à '31 décembre 1999'.

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.

QUANTITES_CLIENTS			VENTES_CLIENTS		
ANNEE	NUMBER(4)	ns	ANNEE	NUMBER(4)	ns
MOIS	NUMBER(2)	ns	MOIS	NUMBER(2)	ns
CODE_CLIENT	CHAR(5)	ns	CODE_CLIENT	CHAR(5)	ns
QUANTITE	NUMBER(15)	ns	VENTE	NUMBER(12,2)	ns
PORT	NUMBER(12,2)	ns	REMISE	NUMBER(12,2)	ns

VENTES_ANNEES			VENTES_MOIS		
ANNEE	NUMBER(4)	ns	ANNEE	NUMBER(4)	ns
VENTE	NUMBER(12,2)	ns	MOIS	NUMBER(2)	ns
REMISE	NUMBER(12,2)	ns	VENTE	NUMBER(12,2)	ns
			REMISE	NUMBER(12,2)	ns

VENTES_CLIENTS_1999			VENTES_CLIENTS_1998		
MOIS	NUMBER(2)	ns	MOIS	NUMBER(2)	ns
CODE_CLIENT	CHAR(5)	ns	CODE_CLIENT	CHAR(5)	ns
VENTE	NUMBER(12,2)	ns	VENTE	NUMBER(12,2)	ns
REMISE	NUMBER(12,2)	ns	REMISE	NUMBER(12,2)	ns

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;

91 ligne(s) créée(s).

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

9 ligne(s) créée(s).

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

77 ligne(s) créée(s).

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 );

2155 ligne(s) créée(s).

SQL> COMMIT;

Validation effectuée.
```

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);
```

91 lignes fusionnées.

```
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);
```

9 lignes fusionnées.

```
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);
```

77 lignes fusionnées.

```
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);

```

2155 lignes fusionnées.

SQL> COMMIT;

Validation effectuée.

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;

0 ligne(s) supprimée(s).

SQL> DELETE VENTES_CLIENTS;

0 ligne(s) supprimée(s).

```

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;

```

1274 ligne(s) créée(s).

```
SQL> COMMIT;
```

Validation effectuée.

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_1996
- VENTES_CLIENTS_1997
- VENTES_CLIENTS_1998

Validez les modifications effectuées.

```

SQL> INSERT ALL
2   WHEN G = 3 THEN
3       INTO VENTES_ANNEES
4       VALUES ( ANNEE, VENTE, REMISE)
5   WHEN G = 1 THEN
6       INTO VENTES_MOIS
7       VALUES ( ANNEE, MOIS, VENTE, REMISE)
8   WHEN ANNEE = 1996 THEN
9       INTO VENTES_CLIENTS_1996
10      VALUES ( MOIS, CODE_CLIENT, VENTE, REMISE)
11  WHEN ANNEE = 1997 THEN
12      INTO VENTES_CLIENTS_1997
13      VALUES ( MOIS, CODE_CLIENT, VENTE, REMISE)
14  WHEN ANNEE = 1998 THEN
15      INTO VENTES_CLIENTS_1998
16      VALUES ( MOIS, CODE_CLIENT, VENTE, REMISE)
17  SELECT GROUPING_ID( EXTRACT ( YEAR  FROM DATE_COMMANDE),
18                      EXTRACT ( MONTH FROM DATE_COMMANDE),
19                      CODE_CLIENT ) G,
20      EXTRACT ( YEAR  FROM DATE_COMMANDE) ANNEE,
21      EXTRACT ( MONTH FROM DATE_COMMANDE) MOIS,
22      CODE_CLIENT,
23      SUM(QUANTITE*PRIX_UNITAIRE) VENTE,
24      SUM(QUANTITE*PRIX_UNITAIRE*REMISE) REMISE
25  FROM  COMMANDES NATURAL JOIN DETAILS_COMMANDES
26  GROUP BY ROLLUP
27      (EXTRACT ( YEAR  FROM DATE_COMMANDE),
28      EXTRACT ( MONTH FROM DATE_COMMANDE),
29      CODE_CLIENT);

```

689 ligne(s) créée(s).

```
SQL> COMMIT;
```

Atelier 14 Les transactions



Questions

14-1.

Réponse : Non

14-2.

Réponse : Oui

14-3.

Réponse : A

14-4.

Réponse : Non

14-5.

Réponse : A, F, G

14-6.

Réponse : B, C, D, E

14-7.

Réponse : 6 000

14-8.

Réponse : 8 000

14-9.

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      ( 10,'Boissons non alcoolisées',
3          'Boissons non alcoolisées' );

```

1 ligne créée.

```

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

```

1 ligne créée.

```

SQL> SAVEPOINT POINT_REPERE_1;

```

Savepoint créé.

Attribuez les produits 1 et 43 à la catégorie numéro 10 et insérez un point de sauvegarde POINT_REPERE_2.

```

SQL> UPDATE PRODUITS SET CODE_CATEGORIE = 10
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 = 11
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 = 11
      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 (10,11);
```

CODE_CATEGORIE	NOM_CATEGORIE	DESCRIPTION
10	Boissons non alcoolisées	Boissons non alcoolisées
11	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 15 La création des tables

Questions



15-1.

Réponse : A, B

15-2.

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

15-3.

Réponse : F

15-4.

Réponse : Oui

15-5.

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 16 La gestion des tables



Questions

16-1.

Réponse : De type « **NOT NULL** »

16-2.

Réponse : D

16-3.

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

16-4.

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

16-5.

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

16-6.

Réponse : A, C

16-7.

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

16-8.

Réponse : Les colonnes supprimées ne peuvent pas être récupérées.

16-9.

Réponse : NON

16-10.

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

16-11.

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

16-12.

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

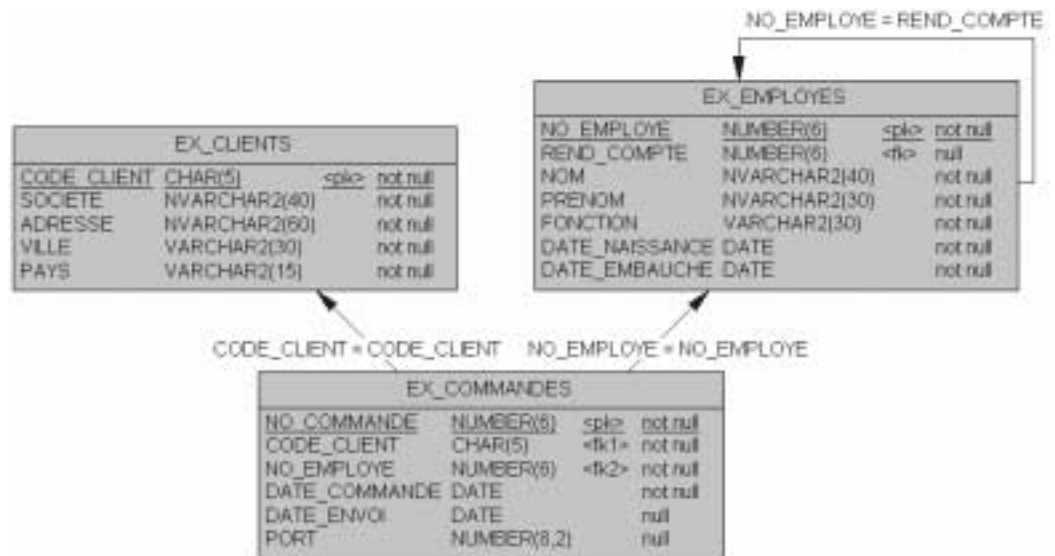
Exercice n° 1 Les contraintes

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_Eмбаuche** ». 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_EмбаУCHE          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_EмбаУCHE CHECK
14        (((DATE_EмбаУCHE - 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). Après 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 17 Les index



Questions

17-1.

Réponse : B-Tree

17-2.

Réponse : Bitmap

17-3.

Réponse : C

Exercice n°1 La création des index

Pour les trois tables créées dans le module précédent créez les index suivants :

- Pour la table EX_CLIENTS, un index B-Tree sur la colonne SOCIETE.
- Pour la table EX_PERSONNES, un index B-Tree pour la clé étrangère.
- Pour la table EX_COMMANDES, créez deux index bitmap pour les deux clés étrangères.

Stockez les index dans le tablespace « GEST_INDX ».

```
SQL> DROP INDEX FK_EX_CLIENTS_COMMANDES;  
SQL> DROP INDEX FK_EX_EMPLOYES_COMMANDES;  
SQL> DROP INDEX FK_EX_EMPLOYES_REND_COMPTE;  
SQL> DROP INDEX EX_EMPLOYES_SOCIETE;  
SQL> CREATE INDEX FK_EX_CLIENTS_COMMANDES ON  
2                                EX_COMMANDES (CODE_CLIENT);  
  
SQL> CREATE INDEX FK_EX_EMPLOYES_COMMANDES ON  
2                                EX_COMMANDES (NO_EMPLOYE);  
  
SQL> CREATE INDEX FK_EX_EMPLOYES_REND_COMPTE ON  
2                                EX_PERSONNES (REND_COMPTE);  
  
SQL> CREATE INDEX EX_EMPLOYES_SOCIETE ON  
2                                EX_CLIENTS (SOCIETE);
```

Pour la table EX_COMMANDES, créez un index qui empêche de saisir deux commandes pour le même client deux fois dans une journée.

```
SQL> CREATE UNIQUE INDEX UNK_EX_COMMANDES_CLIDATE ON  
2                                EX_COMMANDES (CODE_CLIENT, DATE_COMMANDE);
```

Index créé.

Exercice n°2 Les tables organisées en index

Créez une table organisée en index avec la même description que la table DETAILS_COMMANDES. Créez également deux index sur les colonnes REF_PRODUIT et NO_COMMANDE.

```
SQL> DROP TABLE EX_DETAILS_COMMANDES;
```

Table supprimée.

```
SQL> CREATE TABLE EX_DETAILS_COMMANDES (
2      NO_COMMANDE    NUMBER(6,0)  NOT NULL ENABLE,
3      REF_PRODUIT    NUMBER(6,0)  NOT NULL ENABLE,
4      PRIX_UNITAIRE  NUMBER(8,2)  NOT NULL ENABLE,
5      QUANTITE       NUMBER(5,0)  NOT NULL ENABLE,
6      REMISE         FLOAT(126)   NOT NULL ENABLE,
7      CONSTRAINT PK_EX_DET_COMM PRIMARY KEY
8                      ( NO_COMMANDE, REF_PRODUIT ) ENABLE,
9      CONSTRAINT FK_EX_DET_COMM_COMM FOREIGN KEY ( NO_COMMANDE )
10     REFERENCES EX_COMMANDES ( NO_COMMANDE ) ENABLE
11 ) ORGANIZATION INDEX TABLESPACE GEST_INDX;
```

Table créée.

```
SQL> CREATE INDEX FK_EX_COM_DET_COM ON
2      EX_DETAILS_COMMANDES ( NO_COMMANDE ASC )
3      TABLESPACE GEST_INDX;
```

Index créé.

```
SQL> CREATE INDEX FK_EX_PROD_DET_COM ON
2      EX_DETAILS_COMMANDES ( REF_PRODUIT ASC )
3      TABLESPACE GEST_INDX;
```

Index créé.

Atelier 18 Les vues et autres objets



Questions

18-1. :

Réponse : La modification des vues en lecture seule.

18-2.

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 19 Les profils

Questions



19-1.

Réponse : A Le paramètre « **PASSWORD_LOCK_TIME** » indique le temps en jours pendant lequel l'utilisateur ne peut pas se connecter.

Exercice n°1 La création d'un profil

Créez un profil « **APP_PROF** » qui verrouille l'utilisateur après trois échecs de connexion et le maintient ainsi indéfiniment. Un changement de mot de passe est demandé tous les soixante jours. Un ancien mot de passe ne peut pas être réutilisé avant cent vingt jours.

Le profil doit limiter le nombre de sessions par utilisateur à deux.

```
SQL> CREATE PROFILE APP_PROF
2  LIMIT
3      FAILED_LOGIN_ATTEMPTS      3
4      PASSWORD_LIFE_TIME         60
5      PASSWORD_REUSE_TIME        120
6      PASSWORD_LOCK_TIME         UNLIMITED
7      SESSIONS_PER_USER          2;
```

Profil créé.

Atelier 20 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 OBSOLETE_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/OBSOLETE_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 21 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** ».

Atelier 22 Le dictionnaire de données



Questions

22-1.

Réponse : A

22-2.

Réponse : D

Exercice n°1 L'interrogation du dictionnaire de données

Créez une requête qui interroge la vue du dictionnaire de données « **DICTIONARY** ». Elle doit utiliser une variable de substitution pour récupérer uniquement les enregistrements qui correspondent. Le filtre porte sur le nom ou une partie du nom d'une ou plusieurs vues du dictionnaire de données.

```
SQL> SELECT TABLE_NAME, COMMENTS FROM DICT
      2 WHERE TABLE_NAME LIKE '%&NOM%'
Entrez une valeur pour nom : DBA_TABLES
ancien      2 : WHERE TABLE_NAME LIKE '%&NOM%'
nouveau    2 : WHERE TABLE_NAME LIKE '%DBA_TABLES%'
```

TABLE_NAME

COMMENTS

DBA_TABLES

Description of all relational tables in the database

DBA_TABLESPACES

Description of all tablespaces

DBA_TABLESPACE_GROUPS

Description of all tablespace groups

DBA_TABLESPACE_USAGE_METRICS

Description of all tablespace space usage metrics

Exercice n°2 La recherche des données de la base

Affichez l'ensemble des utilisateurs de la base de données ainsi que la date de création de leurs comptes.

```
SQL> SELECT USERNAME, CREATED FROM DBA_USERS;
```

USERNAME

CREATED

SYSTEM

09/06/06

SYS

09/06/06

STAGIAIRE

18/07/06

OLAPSYS	09/06/06
SI_INFORMTN_SCHEMA	09/06/06
MGMT_VIEW	09/06/06
ORDPLUGINS	09/06/06
WKPROXY	09/06/06
XDB	09/06/06
SYSMAN	09/06/06
HR	11/07/06
OE	11/07/06
DIP	09/06/06
OUTLN	09/06/06
SH	11/07/06
ANONYMOUS	09/06/06
CTXSYS	09/06/06

Affichez l'ensemble tables d'un utilisateur de la base de données ainsi que l'emplacement des ces tables.

```
SQL> SELECT TABLE_NAME, TABLESPACE_NAME
2 FROM ALL_TABLES
3 WHERE OWNER like UPPER('%&utilisateur%');
```

TABLE_NAME	TABLESPACE_NAME
CATEGORIES	GEST_DATA
...	

Affichez les index, les champs des index et l'emplacement des index d'un utilisateur de la base de données.

```
SQL> SELECT INDEX_NAME, TABLESPACE_NAME, COLUMN_NAME
2 FROM ALL_INDEXES JOIN ALL_IND_COLUMNS
3 USING( INDEX_NAME, TABLE_NAME)
4 WHERE OWNER LIKE UPPER('%&utilisateur%');
```

INDEX_NAME	TABLESPACE_NAME	COLUMN_NAME
PK_CATEGORIES	GEST_INDX	CODE_CATEGORIE
...		

Octroyez à l'utilisateur « **APP3** » créé dans le module précédent le privilège « **SELECT** » sur l'ensemble des tables du schéma « **STAGIAIRE** ».

```
SQL> SET HEADING OFF
SQL> SET ECHO OFF
SQL> SET FEEDBACK OFF
SQL> SET PAGESIZE 0
SQL> SET LINESIZE 200
SQL> SPOOL C:\GRANT_PUBLIC.SQL
SQL> SELECT 'GRANT SELECT ON ' || TABLE_NAME || ' TO
2 FROM USER_TABLES;
GRANT SELECT ON CATEGORIES TO APP3;
...
SQL> SPOOL OFF
SQL> @C:\GRANT_PUBLIC.SQL
```

Créez pour toutes les tables du schéma « **STAGIAIRE** » des synonymes pour l'utilisateur « **APP3** ».

```
SQL> SET HEADING OFF
```

```
SQL> SET ECHO OFF
SQL> SET FEEDBACK OFF
SQL> SET PAGESIZE 0
SQL> SET LINESIZE 200
SQL> SPOOL C:\GRANT_PUBLIC.SQL
SQL> SELECT 'DROP SYNONYM APP3.' ||
2         TABLE_NAME || ' ;' " --"
3 FROM USER_TABLES;
DROP SYNONYM APP3.CATEGORIES ;
...
SQL> SELECT 'CREATE SYNONYM APP3.' || TABLE_NAME || ' FOR ' ||
2         TABLE_NAME || ' ;' " --"
3 FROM USER_TABLES;
CREATE SYNONYM APP3.CATEGORIES FOR CATEGORIES;
...
SQL> SPOOL OFF
SQL> @C:\GRANT_PUBLIC.SQL
```