Burkina Faso

Unité- Progrès-Justice

ORACLE DATABASE : SQL/SQL*PLUS ➔ 9 h
ORACLE DEVELOPER : PL/SQL ➔9 h
Devoir 1 : ➔ 1 h 30 min
ORACLE DEVELOPER : FORMS & REPORTS ➔ 9 h
Devoir 2 : ➔ 1 h 30 min

Durée : 30 Heures

Enseignante :
Madame IMA RASMATA
Email : ima_ramatou@yahoo.fr
Tel : 00226 70 73 38 41

Période :

Avril – Mai 2021

Burkina Faso

Unité- Progrès-Justice

ORACLE DATABASE : ORACLE SQL et l'environnement SQL*plus

Durée : 9 Heures

Enseignante :

Madame IMA RASMATA

Période :

Avril – Mai 2021

PROGRAMME

1. INTRODUCTION

- 1.1. DÉFINITIONS
- 1.2. L'OFFRE ORACLE
- 1.3. LES COMMANDES
- 1.4. LES OBJETS

2. INTERROGATION DES DONNÉES

- 2.1. SYNTAXE DU VERBE SELECT
- 2.2. INDÉPENDANCE LOGIQUE EXTERNE
- 2.3. ELIMINATION DE DOUBLONS : DISTINCT
- 2.4. OPÉRATION DE SÉLECTION
  - 2.4.1. Opérateurs arithmétiques
  - 2.4.2. Critères de comparaison : opérateurs sur les chaînes : LIKE
  - 2.4.4. Critères de comparaison avec l'opérateur BETWEEN
  - 2.4.5. Critères de comparaison avec une valeur nulle
  - 2.4.6. Les opérateurs ANY, SOME et ALL

PROGRAMME

2.5. EXPRESSIONS ET FONCTIONS

2.5.1. Les expressions

2.5.2. Les fonctions

2.6. LES FONCTIONS DE GROUPE / UTILISATION DE FONCTIONS AGGRÉGATIVES

2.7. PRÉSENTATION DU RÉSULTAT TRIÉ SELON UN ORDRE PRÉCIS

2.8. REQUÊTES MULTI-RELATIONS SANS SOUS-REQUÊTES : LA JOINTURE OU PRODUIT CARTÉSIEN

2.9. REQUÊTES MULTI-RELATIONS AVEC LES OPÉRATEURS ENSEMBLISTES

2.10. REQUÊTES MULTI-RELATIONS AVEC LES OPÉRATEURS ENSEMBLISTES

2.11. SOUS-INTERROGATIONS NON SYNCHRONISÉE

2.12. LA JOINTURE EXTERNE

2.13. LE PARTITIONNEMENT

PROGRAMME

3. MISE À JOUR DES DONNÉES

- 3.1. INSERTION DE LIGNES
- 3.2. MODIFICATION DE LIGNES
- 3.3. SUPPRESSION DE LIGNES
  - 3.3.1. VIA LA COMMANDE DELETE
  - 3.3.2. VIA LA COMMANDE TRUNCATE

4. LE SCHÉMA DE DONNÉES

- 4.1. DÉFINITION DU SCHÉMA : ASPECTS STATIQUES
  - 4.1.1. Les types de données Oracle
  - 4.1.2. Création d'une table
  - 4.1.3. Création d'un index

PROGRAMME

4.2. DÉFINITION DU SCHÉMA : ASPECTS DYNAMIQUES

4.2.1. Modification d'une table

4.3. LE DICTIONNAIRE DE DONNÉES

4.4. AUTRES OBJETS

5. CONCURRENCE D'ACCÈS

5.1. TRANSACTION

5.2. NOTION DE VERROUS

6. LE SCHÉMA EXTERNE (LES VUES)

6.1. DÉFINITION DU SCHÉMA EXTERNE

6.2. MANIPULATION SUR LES VUES

PROGRAMME

7. L'environnement SQL*PLUS

7.1 Les variables de substitution

7.1.1 UTILISATION D'ESPERLUETTE &

7.1.2 SUBSTITUTION DE CHAÎNES DE CARACTÈRES ET DE DATES

7.1.3 UTILISATION DE DOUBLE ESPERLUETTE &&

7.2 Définition des variables de substitution

7.2.1 UTILISATION DE LA COMMANDE DEFINE

7.2.2 UTILISATION DE LA COMMANDE UNDEFINE

7.2.3 UTILISATION DE LA COMMANDE VERIFY

7.3 Personnalisation de l'environnement SQL*PLUS

7.3.1 UTILISATION DE LA COMMANDE SET

7.3.2 LES COMMANDES DE FORMATAGE

7.3.3 UTILISATION DE LA COMMANDE COLUMN

7.3.4 UTILISATION DE LA COMMANDE BREAK

7.3.5 UTILISATION DES COMMANDES TTITLE ET BTITLE

7.3.6 EXÉCUTION DES RAPPORTS FORMATÉS

1.INTRODUCTION

1.1. Définitions

Une base de données est un ensemble d'informations structurées.

Un SGBDR (Système de Gestion de Bases de Données Relationnel) est un logiciel qui permet de :

- stocker,
- consulter,
- modifier,
- supprimer

les données de la base de données.

Un SGBDR stocke les informations dans des tables.

1.1. Définitions (suite ...)

SQL (Structured Query Language) :

- est le langage utilisé pour accéder aux données d'une base de données.

- est normalisé. C'est un standard adopté par l'ANSI (American National Standards Institute).

ANSI SQL89

- est un langage ensembliste (non procédural)

- est un langage « universel » utilisé par :

    * les administrateurs

    * les développeurs

    * les utilisateurs

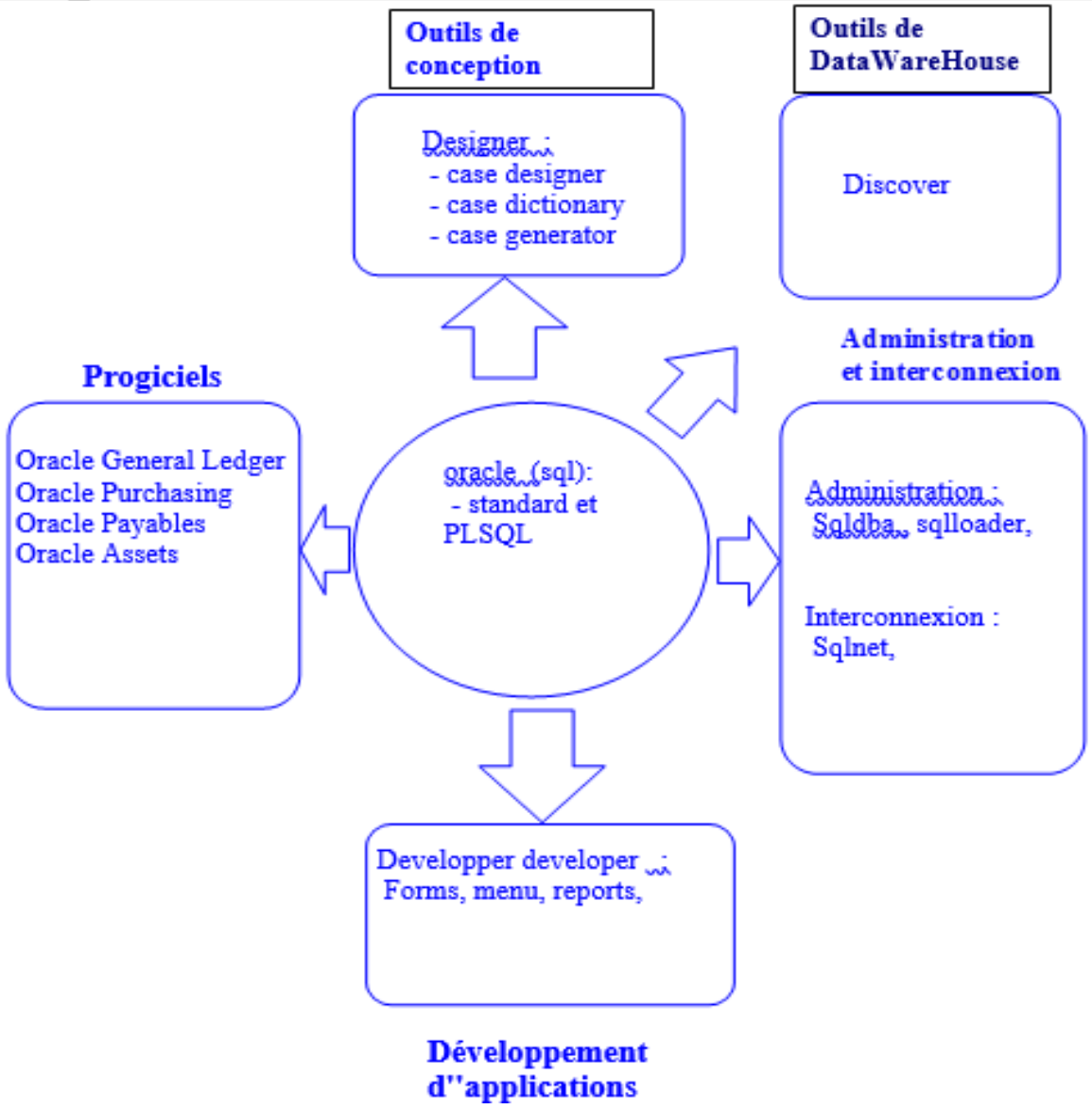
  pour :

    * administrer et contrôler

    * définir et développer

    * manipuler

1.2. L'offre ORACLE



1.3. Les commandes

Commandes de manipulation des données :

- SELECT : interrogation
- INSERT : insertion
- UPDATE : mise à jour
- DELETE : suppression

Les commandes de définition de données :

- CREATE : création d'un objet
- ALTER : modification d'un objet
- TRUNCATE : supprimer les lignes d'une table
- DROP : supprimer un objet
- RENAME : renommer un objet

Remarque : les commandes GRANT et REVOKE seront vues dans le cours d'administration.

1.3. Les commandes

Commandes de manipulation des données :

- SELECT : interrogation
- INSERT : insertion
- UPDATE : mise à jour
- DELETE : suppression

Les commandes de définition de données :

- CREATE : création d'un objet
- ALTER : modification d'un objet
- TRUNCATE : supprimer les lignes d'une table
- DROP : supprimer un objet
- RENAME : renommer un objet

Remarque : les commandes GRANT et REVOKE seront vues dans le cours d'administration.

1.4. Les objets

Les objets du SGBD Relationnel ORACLE sont les suivants :

- les Tables,
- les Vues,
- les Index,
- les Séquences,
- les Synonymes,
- les Clusters.

Seuls les objets TABLES, VUES, INDEX et SYNONYMES seront vus dans ce cours.

2. Interrogation des données

2.1. Syntaxe du verbe SELECT

```
SELECT   [ALL | DISTINCT]  {[schéma.table].*
                  | expr [c_alias], ...}
FROM   [schéma].obj [t_alias], [schéma].obj [t_alias], ...
[WHERE <condition>]
[GROUP BY expr, expr, ...
      [HAVING <condition>]]

[ORDER BY {expr|pos} [ASC|DESC],
                  [{expr|pos} [ASC|DESC], ...]
```

2.1. Syntaxe du verbe SELECT (suite ...)

La clause :

SELECT ...

FROM ...

WHERE ...

est une traduction simple du langage naturel. Elle permet de rechercher les données dans la base dans une ou plusieurs tables, dans une ou plusieurs vues.

Notes :

| : choix entre différentes options

{} : choix obligatoire

[] : facultatif

a) *obj* : peut être une TABLE, une VUE ou un SNAPSHOT

b) *expr* est une expression basée sur les valeurs d'une colonne

c) *c_alias* est le renommage de l'expression

d) *t_alias* est le renommage d'une table, vue ou shnapshot

2.2. Indépendance logique externe (suite ...)

LE RENOMMAGE :

- alias d'attributs et
- alias des tables

Exemple :

```
SQL> SELECT p.pl#  num_pilote
        FROM pilote p;

NUM_PILOTE
------------------
         1
         2
         3
         4
         5
         .
16 ligne(s) sélectionnée(s).
```

Exemple :

```
SQL> SELECT p.pl#  num_pilote
        FROM pilote p;

NUM_PILOTE
----------
         1
         2
         3
         4
         5
         6
         8
         9
        10
        11
        12
        13
        14
        15
        16
        17

16 ligne(s) sélectionnée(s).
```

2.2. Indépendance logique externe (suite ...)

Exemple :

Ecrire de 3 manières différentes une projection sur toutes les colonnes de la table PILOTE.

SQL> SELECT * FROM pilote;

SQL> SELECT a.* FROM pilote a;

SQL > SELECT pilote.* from pilote;

Même résultat dans tous les cas :

PL#	PLNOM	DNAISS	ADR	TEL	SAL
1	Miranda	16/08/52	Sophia Antipolis	93548254	18009
2	St-exupéry	16/10/32	Lyon	91548254	12300
3	Armstrong	11/03/30	Wapakoneta	96548254	24500
4	Tintin	01/08/29	Bruxelles	93548254	21100
5	Gagarine	12/08/34	Klouchino	93548454	22100
6	Baudry	31/08/59	Toulouse	93548444	21000
8	Bush	28/02/24	Milton	44556254	22000
9	Ruskoi	16/08/30	Moscou	73548254	22000
10	Mathé	12/08/38	Paris	23548254	15000
11	Yen	19/09/42	Munich	13548254	29000
12	Icare	17/12/62	Ithaques	73548211	17000,6
13	Mopolo	04/11/55	Nice	93958211	17000,6
14	Chretien	04/11/45		73223322	15000,6
15	Vernes	04/11/35	Paris		17000,6
16	Tournesol	04/11/29	Bruxelles		15000,6
17	Concorde	04/08/66	Paris		21000,6

16 ligne(s) sélectionnée(s).

2.2. Indépendance logique externe (suite …)

LA REDISPOSITION DES COLONNES (des attributs)

Exemple :

SQL> desc pilote;

Nom	Non renseigné	NULL?	Type
PL#		NOT NULL	NUMBER(4)
PLNOM		NOT NULL	CHAR(12)
DNAISS		NOT NULL	DATE
ADR			CHAR(20)
TEL			CHAR(12)
SAL		NOT NULL	NUMBER(7,2)

```
SQL> SELECT pl#, sal, tel,plnom
          FROM pilote;
PL#       SAL TEL          PLNOM
--- --------- ------------ ------------
  1     18009 93548254     Miranda
  2     12300 91548254     St-exupéry
  3     24500 96548254     Armstrong
  4     21100 93548254     Tintin
  5     22100 93548454     Gagarine
  6     21000 93548444     Baudry
  8     22000 44556254     Bush
  9     22000 73548254     Ruskoi
 10      15000 23548254     Mathé
 11      29000 13548254     Yen
 12   17000,6 73548211      Icare
 13   17000,6 93958211      Mopolo
 14   15000,6 73223322      Chretien
 15   17000,6               Vernes
 16   15000,6               Tournesol
 17   21000,6               Concorde
```

2.2. Indépendance logique externe (suite ...)

LES CONSTANTES

On peut répéter une constante pour chaque ligne ramenée.

Les constantes sont de type numérique ou alphanumérique (entre ' ').

Exemple :

```
SQL> SELECT plnom NOM , 'gagne' GAIN ,
                        sal SALAIRE
          FROM pilote;
```

2.2. Indépendance logique externe (suite ...)

LES CONSTANTES

On peut répéter une constante pour chaque ligne ramenée.

Les constantes sont de type numérique ou alphanumérique (entre ' ').

Exemple :

```
SQL> SELECT plnom NOM , 'gagne' GAIN ,
                        sal SALAIRE
          FROM pilote;
```

Exemple :

SQL> SELECT plnom NOM , 'gagne' GAIN , sal SALAIRE FROM pilote;

```
NOM          GAIN    SALAIRE
------------ ----- ---------
Miranda      gagne     18009
St-exupéry   gagne     12300
Armstrong    gagne     24500
Tintin       gagne     21100
Gagarine     gagne     22100
Baudry       gagne     21000
Bush         gagne     22000
Ruskoi       gagne     22000
Mathé        gagne     15000
Yen          gagne     29000
Icare        gagne    17000,6
Mopolo       gagne    17000,6
Chretien     gagne    15000,6
Vernes       gagne    17000,6
Tournesol    gagne    15000,6
Concorde     gagne    21000,6
```

2.2. Indépendance logique externe (suite ...)

LES CALCULS HORIZONTAUX

Le calcul horizontal fait intervenir une ou plusieurs colonnes d'une même table dans un tuple.

Exemple :

```
SQL> SELECT pl#, sal*12  "SALAIRE MENSUEL"
        FROM pilote;
```

PL#	SALAIRE MENSUEL
1	216108
2	147600
3	294000
4	253200
5	265200
6	252000
8	264000
9	264000
10	180000
11	348000
12	204007,2
13	204007,2
14	180007,2
15	204007,2
16	180007,2
17	252007,2

16 ligne(s) sélectionnée(s).

2.2. Indépendance logique externe (suite ...)

LES CALCULS VERTICAUX

Les calculs verticaux font intervenir les valeurs d'une colonne sur l'ensemble ou un sous-ensemble des tuples ramenés par une requête.

Remarque :

l'alias d'une colonne ou d'une expression sera de 30 caractères max. et sera entre "" si l'alias contient des séparateurs.

Exemple :

```
SQL> SELECT avtype TYPE,
                    SUM(cap) "CAPACITE TOTALE"
          FROM avion
           GROUP BY avtype;

TYPE       CAPACITE TOTALE
---------- ---------------
A300                 1300
A320                  320
B707                  400
B727                  250
Caravelle             300
Concorde              650

6 ligne(s) sélectionnée(s).
```

2.3. Elimination de doublons : DISTINCT

Le mot clé DISTINCT dans la clause SELECT :

- réalise un tri sur les colonnes et
- élimine les doublons.

Exemple :

```
SQL> SELECT DISTINCT avtype FROM avion;

AVTYPE
----------
A300
A320
B707
B727
Caravelle
Concorde
6 ligne(s) sélectionnée(s).
```

Il est possible de faire un DISTINCT de plusieurs colonnes.

Exemple :

SQL> SELECT DISTINCT avtype,cap FROM avion;

AVTYPE	CAP
A300	300
A300	400
A320	320
B707	400
B727	250
Caravelle	300
Concorde	300
Concorde	350

EXERCICES 1

Alias des attributs

Ecrire la requête qui présente tous les pilotes de la compagnie avec le listing suivant:

Numéro Nom Adresse Salaire Mensuel

Redisposition des attributs

Ecrire la requête qui présente tous les pilotes de la compagnie avec le listing suivant

Nom Salaire Mensuel Numéro Adresse

Alias d'une table

Ecrire la requête qui renomme(alias) la relation PILOTE en P dans une requête.

Calculs horizontaux

Ecrire la requête qui calcule la durée d'un vol.

Ecrire une requête qui calcule le salaire annuel SAL_ANN, pour chaque pilote.

Calculs verticaux

Ecrire une requête qui calcule la somme des salaires des pilotes.

Distinct

Donner tous les types d'avions de la compagnie

2.4. Opération de sélection

SELECT ...

FROM ...

WHERE [NOT] prédicat1

[AND|OR]

[NOT] prédicat2 ...

La clause WHERE permet d'effectuer un filtrage de tuples. C'est à dire sélectionner un sous-ensemble de lignes dans les tables.

Seules les lignes vérifiant la clause WHERE seront retournées.

Prédicat :

nom de colonne		nom de colonne
constante	OPERATEUR	constante
expression		expression

2.4. Opération de sélection

Les opérateurs logiques (AND, OR) peuvent être utilisés dans le cas de prédicats multiples.

- L'opérateur NOT inverse le sens du prédicat.

- Pas de limite dans le nombre de prédicats.

2.4. Opération de sélection (suite ...)

Exemples :

Lister tous les pilotes de la compagnie

```
SQL> SELECT *
        FROM pilote;
```

==> - pas de sélection

- tous les tuples de la relation PILOTE sont ramenés

Lister les pilotes qui vivent à Nice

```
SQL> SELECT *
        FROM PILOTE
        WHERE ADR='Nice';
```

==> - sélection : clause WHERE

- seuls les tuples de la relation PILOTE vérifant la clause WHERE sont ramenés

2.4.1. Opérateurs arithmétiques

Dans les critères de la clause WHERE, nous pouvons avoir les opérateurs de comparaison arithmétiques suivants :

= : égal,

!= : différent,

> : supérieur,

>= : supérieur ou égal,

< : inférieur,

<= : inférieur ou égal.

Exemple :

Liste des pilotes qui gagnent plus de 10000 et dont le numéro de tel est 93000000

```
SQL> SELECT plnom
        FROM pilote
        WHERE sal > 10000
            AND  tel='93000000';
```

2.4.2. Critères de comparaison : opérateurs sur les chaînes : LIKE

Opérateur LIKE

Caractères jokers de l'opérateur LIKE :

% : remplace 0 à n caractères

_ : remplace 1 et un seul caractère

Exemple 1 :

Sélectionnez les pilotes dont le nom commence par M.

```
SQL> SELECT *
        FROM pilote
        WHERE plnom LIKE 'M%';
```

Exemple 2 :

Sélectionnez les pilotes dont le nom contient un A en troisième position.

```
SQL> SELECT *  FROM pilote
        WHERE plnom LIKE '___A%';
```

2.4.2. Critères de comparaison : opérateurs sur les chaînes : LIKE

La clause ESCAPE permet de de-spécialiser les caractères jokers :

_

et

%.

Le caractère précisé derrière la clause ESCAPE permet la recherche des caractères _ et % dans une chaîne de caractères.

Exemple 3 :

Sélectionnez les pilotes dont le nom contient le caractère _.

```
SQL> SELECT *
        FROM pilote
        WHERE plnom LIKE '%*_%' ESCAPE '*';
```

2.4.3. Critères de comparaison avec l'opérateur IN

IN est l'opérateur qui permet de tester l'appartenance de la valeur d'une colonne à une liste.

Exemples :

Liste des vols dont la ville d'arrivée est Nice ou Paris.

```
SQL> SELECT vol#
        FROM vol
        WHERE va IN ('Nice ', 'Paris');
```

2.4.4. Critères de comparaison avec l'opérateur BETWEEN

BETWEEN est l'opérateur qui permet de tester si une valeur appartient à un intervalle.

Remarque : les bornes sont incluses.

Exemple :

Salaire et nom des pilotes gagnant entre 15000 et 18000

```
SQL> SELECT plnom, sal
        FROM pilote
        WHERE sal BETWEEN 15000 AND 18000;

PLNOM            SAL
---------- ---------
Mathé           15000
Icare         17000,6
Mopolo        17000,6
Chretien      15000,6
Vernes        17000,6
Tournesol     15000,6

6 ligne(s) sélectionnée(s).
```

2.4.5. Critères de comparaison avec une valeur nulle

IS NULL et IS NOT NULL sont les opérateurs qui permettent de tester si une valeur a été définie ou pas pour une colonne.

NULL : non défini.

SELECT ...

FROM table

WHERE coli **IS NULL**; coli non renseignée

ou SELECT ...

FROM table

WHERE coli **IS NOT NULL**; coli renseignée

Remarque : pour tester l'absence de valeur , ne pas utiliser = NULL ou != NULL.

Note : la syntaxe de comparaison est la suivante :

colonne IS NULL | IS NOT NULL

Exemple :

Nom des pilotes dont le numéro de tél. n'est pas renseigné

```
SQL> SELECT plnom
        FROM pilote
        WHERE tel IS NULL;
```

2.4.6. Les opérateurs ANY, SOME et ALL

Ils se combinent avec l'un des opérateurs arithmétiques :

{= | != | > | >= | < | <= } ANY : au moins 1 ...

SOME : au moins 1 ...

ALL : tout ...

Exemple 1 :

Sélectionnez les pilotes dont l'adresse est 'Nice' ou 'Paris'

```
SQL> SELECT plnom
        FROM pilote
        WHERE adr = ANY ('Nice', 'Paris');
```

Remarques :

- l'opérateur **ANY** est équivalent à l'opérateur **SOME**.

- la condition **=ANY** est équivalent à l'opérateur **IN**.

2.4.6. Les opérateurs ANY, SOME et ALL (suite ...)

Exemple 2 :

Sélectionnez les pilotes dont le salaire n'est pas un nombre rond.

```
SQL> SELECT plnom
        FROM pilote
        WHERE sal != ALL (12000, 13000, 14000, 15000, 16000, 17000, 18000, 19000, 20000, 21000,
22000, 24000, 25000, 26000, 27000, 28000,29000);
```

Remarque :

La condition **!= ALL** est équivalente à la condition **NOT IN**.

EXERCICES 2

"Numéros et type d'avions de capacité supérieure à 300"

"Nom des pilotes habitants Nice ou Paris"

"Quels sont les noms de pilotes comportant un 't' en quatrième position ou dont le nom se prononce 'Bodri'.

"Quels sont les vols au départ de Nice, Paris ou Bordeaux ?"

"Quels sont les avions dont la capacité est comprise entre 250 et 310 ?"

"Quels sont les pilotes dont l'adresse ou le téléphone sont inconnus ?"

"Nom des pilotes ayant un 'a' et un 'e' dans leur nom"

"Nom des pilotes ayant 2 'o' dans leur nom "

"Nom des pilotes dont le numéro de téléphone est renseigné"

EXERCICES 2

"Numéros et type d'avions de capacité supérieure à 300"

"Nom des pilotes habitants Nice ou Paris"

"Quels sont les noms de pilotes comportant un 't' en quatrième position ou dont le nom se prononce 'Bodri'.

"Quels sont les vols au départ de Nice, Paris ou Bordeaux ?"

"Quels sont les avions dont la capacité est comprise entre 250 et 310 ?"

"Quels sont les pilotes dont l'adresse ou le téléphone sont inconnus ?"

"Nom des pilotes ayant un 'a' et un 'e' dans leur nom"

"Nom des pilotes ayant 2 'o' dans leur nom "

"Nom des pilotes dont le numéro de téléphone est renseigné"

2.5. Expressions et fonctions

L'objectif est de faire des calculs sur des :

- constantes,
- variables

de type :

- numériques,
- caractères,
- dates.

2.5.1. Les expressions

Colonne Colonne
Constante Opérateur Constante
Fonction Fonction

- Opérateurs arithmétiques : + - * /

- Opérateur sur chaînes de caractères : ||

- Pas d'opérateurs spécifiques aux dates.

Exemple1 :

Simuler une augmentation de 10% des salaires des pilotes

```
SQL> SELECT sal * 1.10 AUGMENTATION
        FROM pilote;
```

2.5.1. Les expressions

Colonne Colonne
Constante Opérateur Constante
Fonction Fonction

- Opérateurs arithmétiques : + - * /

- Opérateur sur chaînes de caractères : ||

- Pas d'opérateurs spécifiques aux dates.

Exemple1 :

Simuler une augmentation de 10% des salaires des pilotes

```
SQL> SELECT sal * 1.10 AUGMENTATION
        FROM pilote;
```

2.5.1. Les expressions (suite ...)

Exemple 3 :

ajouter 3 jours à une date

'08-DEC-20' + 3 = '11-DEC-20'

Exemple 4 :

enlever 3 jours à une date

'11-DEC-20' - 3 = '08-DEC-20'

Exemple 5 :

nombre de jours entre 2 dates

date1 - date 2 = nbjours

2.5.1. Les expressions (suite ...)

Exemple 6 :

Noms et adresses des pilotes

```
SQL> SELECT plnom || '----->' || adr
        FROM pilote;
```

```
PLNOM||'---->'||ADR
----------------------------------
Miranda      ---->Sophia Antipolis
St-exupéry   ---->Lyon
Armstrong    ---->Wapakoneta
Tintin       ----->Bruxelles
Gagarine     ---->Klouchino
Baudry       ----->Toulouse
Bush         ----->Milton
Ruskoi       ---->Moscou
Mathé        ----->Paris
Yen          ---->Munich
Icare        ----->Ithaques
Mopolo       ---->Nice
Chretien     ---->
Vernes       ---->Paris
Tournesol    ---->Bruxelles
Concorde     ----->Paris
scott        ----->Nice
Conficius    ----->Pekin
18 ligne(s) sélectionnée(s).
```

5.2. Les fonctions

Fonctions numériques

- ABS(n) : valeur absolue de n
- MOD(m,n) : reste de la division de m par n
- SQRT(n) : racine carrée de n (message d'erreur si n < 0)
- ROUND(n,[m]) : arrondi de n à 10-m

Ex : ROUND(125.2) = 125

ROUND(1600,-3) = 2000

ROUND(1100,- 3) = 1000

ROUND(345.343,2) = 345.34

ROUND(345.347,2) = 345.35

- TRUNC(n,[m]) : n tronqué à 10-m

Ex : TRUNC(2500,-3) = 2000

TRUNC(2400,-3) = 2000

TRUNC(345.343,2) = 345.34

TRUNC(345.347,2) = 345.34

2.5.2. Les fonctions (suite ...)

Fonctions caractères

- LENGTH(chaîne) : longueur de la chaîne

- UPPER(chaîne) : toutes les lettres de la chaîne en majuscules

- LOWER(chaîne) : toutes les lettres de la chaîne en minuscules

- INITCAP(chaîne) : première lettre de chaque mot de la chaîne en majuscules, les autres en minuscules)

- LPAD(chaîne,lg,[chaîne]) : compléter à gauche par une chaîne de caractères sur une longueur donnée.

Exemple :

LPAD('DUPOND',10,'*# ') = '*#*#DUPOND'

- RPAD(chaîne,lg,[chaîne]) : compléter à droite par une chaîne de caractères sur une longueur donnée.

Exemple :

RPAD('DUPOND',10,'* ') = 'DUPOND****'

Remarque : LPAD et RPAD peuvent tronquer une chaîne si lg < longueur totale de la chaîne.

2.5.2. Les fonctions (suite ...)

- LTRIM(chaîne[,caractères]) : suppression à gauche de caractères dans la chaîne.

Exemple :

LTRIM('DUPOND','DU ') = 'POND'

- RTRIM(chaîne[,caractères]) : suppression à droite de caractères dans la chaîne.

Exemple :

RTRIM('DUPOND','UD ') = 'DUPON'

- SUBSTR(chaîne,position[,longueur]) : extraction d'une chaîne à partir d'une position donnée et sur une longueur donnée

Exemple :

SUBSTR('DUPOND',2,3) = 'UPO'

- INSTR(chaîne, sous_chaîne,[,position[,n]]) : recherche de la position de la n ième occurence d'une chaîne de caractères dans une autre chaîne de caractères à partir d'une position données.

Exemple :

INSTR('DUPOND','D',1,2) = 6

2.5.2. Les fonctions (suite ...)

- REPLACE(chaîne,car[,chaîne]) : remplace un ensemble de caractères

Exemples :

REPLACE('TETE','E', 'O') = 'TOTO'

REPLACE('TATA','T') = 'AA'

2.5.2. Les fonctions (suite ...)

- REPLACE(chaîne,car[,chaîne]) : remplace un ensemble de caractères

Exemples :

REPLACE('TETE','E', 'O') = 'TOTO'

REPLACE('TATA','T') = 'AA'

Fonctions date :

- LAST_DAY(date) : dernier jour du mois d'une date donnée
- NEXT_DAY(date, jour) : date du prochain jour à partir d'une date donnée.
- ADD_MONTHS(date,n) : ajoute n mois à une date donnée.
- MONTHS_BETWEEN(date1,date2) : nombre de mois entre 2 dates.
- ROUND(date[,'precision']) : arrondi d'une date en fonction de la précision

Exemples : SYSDATE = '12-JUL-20'

ROUND(sysdate,'MM') = '01-JUL-20'

ROUND(sysdate + 4 ,'MM') = '01-AUG-20'

ROUND(sysdate,'YY') = '01-JAN-21' avec sysdate = 26/04/2021

- TRUNC(date[,'precision']) : truncature d'une date en fonction de la précision

Exemples :

TRUNC(sysdate,'MM') = '01-JUL-20'

TRUNC(sysdate + 4 ,'MM') = '01-JUL-20'

TRUNC(sysdate,'YY') = '01-JAN-21' avec sysdate = 26/04/2021

2.5.2. Les fonctions (suite ...)

Fonctions de conversion de types :

- TO_NUMBER(chaîne) : conversion d'une chaîne de caractères en nombre

Exemple : TO_NUMBER('567') = 567

- TO_CHAR(chaîne[,'format']) : conversion d'une expression (date ou numérique) en chaîne de caractères selon un format de présentation.

- TO_DATE(chaîne[,'format']) : conversion d'une chaîne en date selon un format.

Quelques formats numériques :

9	Affichage de cette valeur si elle est différente de 0
0	Affichage de zéros à gauche pour une valeur à zéro
$	Affichage de la valeur préfixée par le signe '$ '
,	Affichage de ',' à l'endroit indiqué
.	Affichage du point décima à l'endroit indiqué

Exemple : TO_CHAR(1234,'0999999') = 0001234

2.5.2. Les fonctions (suite ...)

Quelques formats de conversion de date :

TO_CHAR(date[,'format'])

FORMAT étant la combinaison de codes suivants:

YYYY	Année
YY	2 derniers chiffres de l'année
MM	numéro du mois
DD	numéro du jour dans le mois
HH	heure sur 12 heures
HH24	heure sur 24 heures
MI	minutes
SS	secondes
...	

Exemple :

```
SELECT
        TO_CHAR(SYSDATE,'DD MM YYYY HH24 : MI')
FROM dual;
```

==> 26 04 2021 15 :30

FIN COURS J 1

DATE : 26/04/2021

Horaire : 15H - 18H

2.5.2. Les fonctions (suite ...)

Pour avoir les dates en lettres utiliser les formats suivants :

YEAR	année en toutes lettres
MONTH	mois en toutes lettres
MON	nom du mois sur 3 lettres
DAY	nom du jour
DY	nom du jour sur 3 lettres
SP	nombre en toutes lettres
...	

Exemple :

TO_CHAR(SYSDATE,' « LE » DD MONTH YYYY « A » HH24 : MI')

==> LE 26 MARS 2021 A 15 : 30

2.5.2. Les fonctions (suite ...)

Fonctions diverses :

NVL(expr,valeur)

==> Si expr IS NULL

Alors valeur

Sinon expr

Finsi

Exemple :

```
SQL> SELECT   NVL(sal,0)
          FROM pilote;
 DECODE(expression, valeur1, result1,
                  [, valeur2, result2]
                  ...
                  [,defaut]
```

==> Si expression = valeur1

Alors result1

Sinon Si expression = valeur2

Alors result2

Sinon defaut

Finsi

Finsi

Remarque : result1, result2, ... defaut peuvent être de types différents.

EXERCICES 3

"Lister les pilotes avec leur salaire tronqués au millier"

*"Lister les pilotes avec leur salaire. Pour ceux gagnat 17000,6 remplacer le salaire par '****' "*

"Sélectionner les pilotes et leur téléphone. Pour ceux dont le télephone n'est pas renseigné, mettre ? "

2.6. Les fonctions de groupe / utilisation de fonctions aggrégatives

Les **fonctions de groupe** sont les suivantes :

- AVG(expr) moyenne
- COUNT(expr) nombre
- MAX(expr) valeur maximim
- MIN(expr) valeur minimum
- STDDEV(expr) écart-type
- SUM(expr) somme
- VARIANCE(expr) variance

Remarques :

- les valeurs NULL sont ignorées.
- COUNT(*) permet de compter les lignes d'une table.

Exemple :

```
SQL> SELECT adr, AVG(sal), COUNT(sal), MAX(sal),
MIN(sal), STDDEV(sal),SUM(sal),
                                VARIANCE(sal)
        FROM pilote GROUP BY adr ;
```

2.7. Présentation du résultat trié selon un ordre précis

Un résultat peut être trié grâce à la clause ORDER BY

- de façon ascendante ASC ou
- descendante DESC.

Remarques :

- par défaut en Oracle le tri est toujours ascendant.
- 16 critères de tri maximum.
- dans un tri par ordre croissant les valeurs NULL apparaissent toujours en dernier

Exemple :

```
SQL> SELECT plnom, adr
        FROM pilote
        ORDER BY plnom;
```

2.8. Utilisation des pseudo colonnes ROWID, USER et SYSDATE

ROWID, USER et SYSDATE représentent respectivement

- l'adresse d'un tuple, composée de trois champs :
    * numéro de bloc dans le fichier,
    * le numéro de tuple dans le bloc et
    * le numéro de fichier),

- l'utilisateur courant d'Oracle

- la date système.

Ce sont entre autres des colonnes implicites de toute table d'Oracle.

Note : la table DUAL appartient à SYS. Elle possède une seule colonne DUMMY et une seule ligne avec pour valeur X. Cette table sert à sélectionner des constantes, des pseudo colonnes ou des expressions en une seule ligne.

Exemple :

```
SQL> select SYSDATE, USER FROM SYS.DUAL;


SYSDATE  USER
-------- ------------------------------
26/04/21 SCOTT
```

EXERCICES Série 4

"Ecrire une requête qui donne le salaire du pilote qui gagne le plus :

<valeur à calculer> "Max salaire Pilote "

"Quels sont les noms, l'adresse et le salaire des pilotes de la compagnie, triés en ordre croissant sur l'adresse, et pour une même adresse en ordre décroissant sur le salaire ? "

"Ecrire une requête qui recherche si l'utilisateur courant d'Oracle est un pilote ?"

"Ecrire une requête qui rend ROWID, USER, SYSDATE, Numéros de vol de tous les vols effectués à la date d'aujourd'hui par le pilote Numéro 4 ?". L'heure de départ et d'arrivée doivent apparaître dans la liste des colonnes de projection.

8. Requêtes multi-relations sans sous-requêtes : la jointure ou produit cartésien

L'objectif de la **jointure** est de ramener sur une même ligne le résultat des informations venant de différentes tables.

<u>Décomposition de la jointure :</u>

1. Sélection
2. Projection des colonnes des différentes tables (colonnes du SELECT + colonnes de jointure)
3. Prédicat de jointure
4. Projection des colonnes du SELECT

Remarques :

- dans le prédicat de jointure comme dans le SELECT, préfixer les attributs si il y a ambigu
- dans le prédicat de jointure, les alias des tables peuvent être utilisés.

L'objectif de l'**auto-jointure** est de ramener sur la même ligne le résultat des informations provenant de 2 lignes de la même table.

2.9. Requêtes multi-relations avec les opérateurs ensemblistes

L'objectif est de manipuler les ensembles ramenés par plusieurs SELECT à l'aide des opérateurs ensemblistes.

Les opérateurs ensemblistes sont :

- l'union : UNION,
- l'intersection : INTERSECT et
- la différence : MINUS

Principe :

SELECT ... FROM ... WHERE ... ==> ensemble

opérateur ensembliste

SELECT ... FROM ... WHERE ... ==> ensemble

opérateur ensembliste

SELECT ... FROM ... WHERE ... ==> ensemble

...

SELECT ... FROM ... WHERE ... ==> ensemble

[ORDER BY]

2.10. Requêtes multi-relations avec les opérateurs ensemblistes (suite ...)

Règles :

- même nombre de variables en projection
- correspondance du type
- colonne de tri référencées par numéro d'ordre

Résultat :

- les titres des colonnes sont ceux du premier SELECT
- la largeur de la colonne est celle de la plus grande largeur parmi les SELECT
- opération distincte implicite (sauf UNION ALL)

2.11. Sous-interrogations non synchronisée

Principe :

Lorsque dans un prédicat un des 2 arguments n'est pas connu, on utilise les sous-interrogations.

SELECT ...

FROM ...

WHERE variable Op **?**

Le **?** n'étant pas connu, il sera le résultat d'une sous-requête.

Règle d'exécution :

c'est la sous-requête de niveau le plus bas qui est évaluée en premier, puis la requête de niveau immédiatement supérieur, ...

CAS 1 : sous-interrogation ramenant une valeur

On utilise les opérateurs =, >, ...

CAS 2 : sous-interrogation ramenant plusieurs valeurs

On utilise les ALL, IN, ANY, SOME.

Remarque :

une sous-interrogation peut ramener plusieurs colonnes. (on teste l'égalité ou l'inégalité).

2.12. La jointure externe

La jointure externe ("outer join") permet de ramener sur la même ligne des informations venant de plsuieurs tables ainsi que les lignes d'une des tables n'ayant pas de correspondan

Cette fonctionnalité est directement offerte par SQL d'Oracle, en faisant suivre, dans la condition de jointure, la colonne de la table dont on veut les lignes non sélectionnées, par signe (+). Ce signe signifie qu'un tuple supplémentaire ne contenant que des blancs (une valeur NULL dans chaque colonne) a été ajouté à la table concernée lorsque la requête a été lancée. Ce tuple "NULL" est ensuite joint aux tuples de la seconde table, ce qui permet de visualiser aussi les tuples non sélectionnés.

Le (+) est à mettre du côté de la clé étrangère puisque c'est uniquement de ce que côté qu'il peut ne pas y avoir de correspondant pour des clés primaires.

Contraintes :

on ne peut effectuer une jointure externe entre plus de deux tables dans une même clause SELECT. Autrement, l'opérateur de jointure externe (+) doit apparaître au plus une fois dans un predicat.

2.13. Le partitionnement

Le partitionnement permet de regrouper les lignes résultat en fonction des différentes valeurs prises par une colonne spécifiée.

```
SELECT     ...
FROM       <Nom_table> , ...
GROUP BY   <Colonne> [, <colonne>, ...]
[HAVING <condition>] ;
```

La spécification de la clause GROUP BY entraîne la création d'autant de sous-tables qu'il y a de valeurs différentes pour la colonne de partitionnement spécifiée.

De même que la clause WHERE joue le rôle de filtre pour la clause SELECT, la clause HAVIN joue le rôle de filtre pour la clause GROUP BY. L'exécution de la clause HAVING sera effect juste après celle du GROUP BY, pour sélectionner les sous-tables qui satisfont la condition spécifiée.

Contraintes :

- la colonne de partitionnement doit figurer dans la clause SELECT.

- un seul GROUP BY est autorisé par requête.

-pas de GROUP BY dans une sous-requête.

EXERCICES Série 7

"Pour chaque ville de localisation d'avions de la compagnie (sauf "Paris") donner le nombr
les capacités minimales et maximales d'avions qui s'y trouvent ?"

"Quels sont les pilotes (avec leur nombre de vols) parmi les pilotes N° 1, 2, 3 , 4 et 13 qu
assurent au moins 2 vols ?"

"Quelle est la capacité moyenne des avions par ville et par type ? "

3. Mise à jour des données

L'objectif de ce chapitre est de se familiariser avec les commandes de mise à jour des données d'une base.

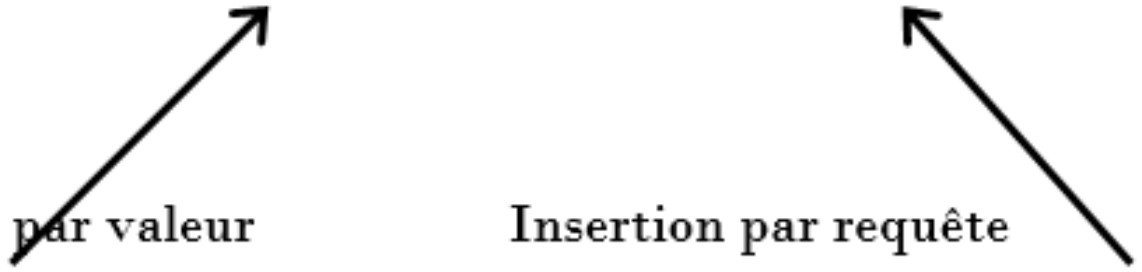
Commandes :

- d'insertion (INSERT),
- de suppression (DELETE) et
- de mise à jour (UPDATE)

des données dans une base Oracle.

3.1. Insertion de lignes

INSERT INTO
<nom_user.nom_table | nom_user.nom_vue>
[(nom_colonnes[,nom_colonnes])]
VALUES (valeurs[,valeurs]) | sous_requête ;



3.1. Insertion de lignes

Remarque :
si toutes les valeurs des colonnes de la table sont insérées, il est inutile de préciser les colonnes. Si seules quelques valeurs sont insérées, préciser les colonnes.

Exemples :

```
SQL> insert into  pilote(pl#,plnom,dnaiss,sal)
        values(2, 'St-exupéry', '16/10/32', 12300.0);

SQL> insert into avion
        values(7, 'Mercure', 300, 'Paris', 'En service');

SQL> insert into vol2
        select * from vol
        where vd='Paris';
```

3.2. Modification de lignes

UPDATE <[nom_user].nom_table | nom_vue>

SET nom_colonne1 = <expression1 | ordre_select>
[, nom_colonne2 = <expression | ordre_select> ...]

WHERE <critères_de_qualification>;

Exemple :

Augmenter les pilotes habitant Nice de 10%

3.2. Modification de lignes

UPDATE <[nom_user].nom_table | nom_vue>

SET nom_colonne1 = <expression1 | ordre_select>
[, nom_colonne2 = <expression | ordre_select> ...]

WHERE <critères_de_qualification>;

Exemple :

Augmenter les pilotes habitant Nice de 10%

```
SQL> UPDATE pilote
        SET sal = sal *1.10
        WHERE adr='Nice';
```

3.3. Suppression de lignes

3.3.1. Via la commande DELETE

DELETE FROM <nom_table | nom_vue>

[**WHERE** <critère_de_qualification>] ;

Remarque :
si pas de claure WHERE, la table entière est vidée.

Exemples :

Supprimer les pilotes habitant Nice

```
SQL > DELETE FROM pilote
          WHERE adr= 'Nice';
```

Supprimer tous les pilotes

```
SQL > DELETE FROM pilote;
```

3.3.2. Via la commande TRUNCATE

TRUNCATE TABLE nom_table

Cette commande permet d'effectuer des suppressions rapides. C'est une commande du LDD d'Oracle et à ce titre équivaut à un commit.

Exemple :

SQL> TRUNCATE TABLE pilote;

Remarque :
Autre manière de supprimer les données d 'une table :
- la supprimer,
- la recréer

Exemple :
SQL> DROP TABLE pilote;
SQL> CREATE TABLE pilote(...);

3.3.2. Via la commande TRUNCATE (suite ...)

Avantages / Inconvénients des 3 solutions :

1ère option DELETE :
- la suppression avec DELETE consomme de *nombreuses ressources* : espace RedoLog, rollbck segment, ...

- pour chaque ligne supprimée, des *triggers* peuvent se déclencher

- la *place* prise par les lignes de la table n'est pas libérée. Elle reste associée à la table.

2ème option DROP :

- tous les index, contraintes d'intégrité et triggers associés à la table sont égelement supprimés

- tous les GRANT sur cette table sont supprimés

3.3.2. Via la commande TRUNCATE (suite ...)

3ème option TRUNCATE :

- truncate est plus rapide car cette commande ne génère pas d'informations (rollback) permettant de défaire cette suppression. L'ordre est validé (commit) de suite.

- truncate est irréversible pour la même raison.

- les contraintes, triggers et autorisations associés à la table ne sont pas impactés

- l'espace prise par la table et ses index peut être libéré (drop storage)

- les triggers ne sont pas déclenchés

3.3.2. Via la commande TRUNCATE (suite ...)

3ème option TRUNCATE :

- truncate est plus rapide car cette commande ne génère pas d'informations (rollback) permettant de défaire cette suppression. L'ordre est validé (commit) de suite.

- truncate est irréversible pour la même raison.

- les contraintes, triggers et autorisations associés à la table ne sont pas impactés

- l'espace prise par la table et ses index peut être libéré (drop storage)

- les triggers ne sont pas déclenchés

EXERCICES 8

Effectuer des insertions respectivement dans pilote, avion et vol. Vérifier si les contraintes l'intégrités structurelles (entitité, domaine et de référence) sont prises en comptes. Vérifier aussi les valeurs nulles.

Note : insérer un pilote ayant votre nom de login oracle et 2 vols effectués par ce pilote.

Effectuer une insertion dans la table PILOTE2 via une sous-requête sur PILOTE. Mettre à jour le salaire du pilote numéro 3 à 19000 F et Valider.

Supprimer le pilote numéro 11 et invalider.

Supprimer les lignes de la tables PILOTE2 via TRUNCATE. Tentez un ROLLBACK.

4. Le schéma de données

Les chapitres précédents nous ont permis d'aborder l'aspect Manipulation de Données (*LMD*) du langage SQL.

Ce chapitre va nous permettre d'aborder l'aspect définition des données : le Langage de Définition de Données (*LDD*).

4.1. Définition du schéma : aspects statiques

4.1.1. Les types de données Oracle

CHAR(taille) : Chaîne - longueur fixe - de 1 à 255 octets

VARCHAR2(taille) : Chaîne de taille variable 1...2000 bytes

DATE : format par défaut JJ-MON-AA

LONG : type texte (taille jusqu'à 2Gbytes)

RAW(taille) : type binaire (taille de 1 à 255bytes)

LONG RAW : type binaire long (taille jusqu'à 2 Go)

NUMBER(n1[, n2]) :
n1 = nombre de digits du décimal (de 1 à 38)
n2 = nombre de digits après la virgule

ROWID : Chaîne hex. représentant l'adresse unique d'une ligne d'une table.

Remarque : une seule colonne de type LONG ou LONG RAW par table.

4.1.1. Création d'une table

CREATE TABLE <user>.<nom_table>
{(
<def_colonne1>,
<def_colonne2>,
...
<def_colonne2>

[contrainte_table]
)]
| as subquery};

FIN COURS J 2

DATE : 27/04/2021

Horaire : 15H - 18H

ORACLE DATABASE :

ORACLE SQL et l'ENVIRONNEMENT SQL*PLUS

Merci pour votre attention

Questions