SQL Oracle

G. Mopolo-Moké

prof. PAST UNSA

2018 / 2019

Plan Général

- 1. Introduction
  - 1.1 Définition
  - 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
  - 2.5. Expressions et fonctions
  - 2.6. Les fonctions de groupe / utilisation de fonctions aggrégatives
  - 2.7. Présentation du résultat trié selon un ordre précis
  - 2.8. Utilisation des pseudo colonnes
  - 2.9. Requêtes multi-relations sans sous-requêtes : la jointure ou produit cartésien
  - 2.10. Requêtes multi-relations avec les opérateurs ensemblistes

Plan Général

- 2. Interrogation des données (suite)
  - 2.11. Sous-interrogations non synchronisée
  - 2.12. La jointure externe
  - 2.13. Sous-interrogations synchronisée
  - 2.14. La recherche hiérarchique
  - 2.15. Le partitionnement

- 3. Mise à jour des données
  - 3.1. Insertion de lignes
  - 3.2. Modification de lignes
  - 3.3. Suppression de lignes

- 4. Le schéma de données
  - 4.1 Les principaux objets d'une base Oracle
  - 4.2 Les règles de nommage des objets
  - 4.3 Les types de données
  - 4.4 Comparaison des chaînes de caractères
  - 4.5 Création d'une table
  - 4.6 Contraintes d'intégrité

Plan Général

- 4. Le schéma de données **(suite)**
  - 4.7 Création d'un index
  - 4.8 Modification d'une table
  - 4.9 Définition des commentaires
  - 4.10 Consultation de la structure d'une table
  - 4.11 Création d'un synonyme
  - 4.12 Les séquences
  - 4.13 Le dictionnaire de données d'Oracle

- 5. Concurrence d'accès
  - 5.1 Transaction
  - 5.2 Gestion des verrous

- 6. Les vues
  - 6.1. Création d'une vue
  - 6.2. Manipulation sur les vues

1. Introduction

1.1 Définition

- **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,
  - sécuriser
  - consulter,
  - modifier,
  - supprimer

  les données d'une base de données

- **Un SGBDR stocke les informations dans des tables.**

1.1 Définition

- **SQL** (Strutured Query Language) :

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

  - **Est un langage normalisé.** C'est un standard adopté par l'ANSI (American National Standards Institute).
    ANSI SQL92, SQL99, SQL2003, SQL2007

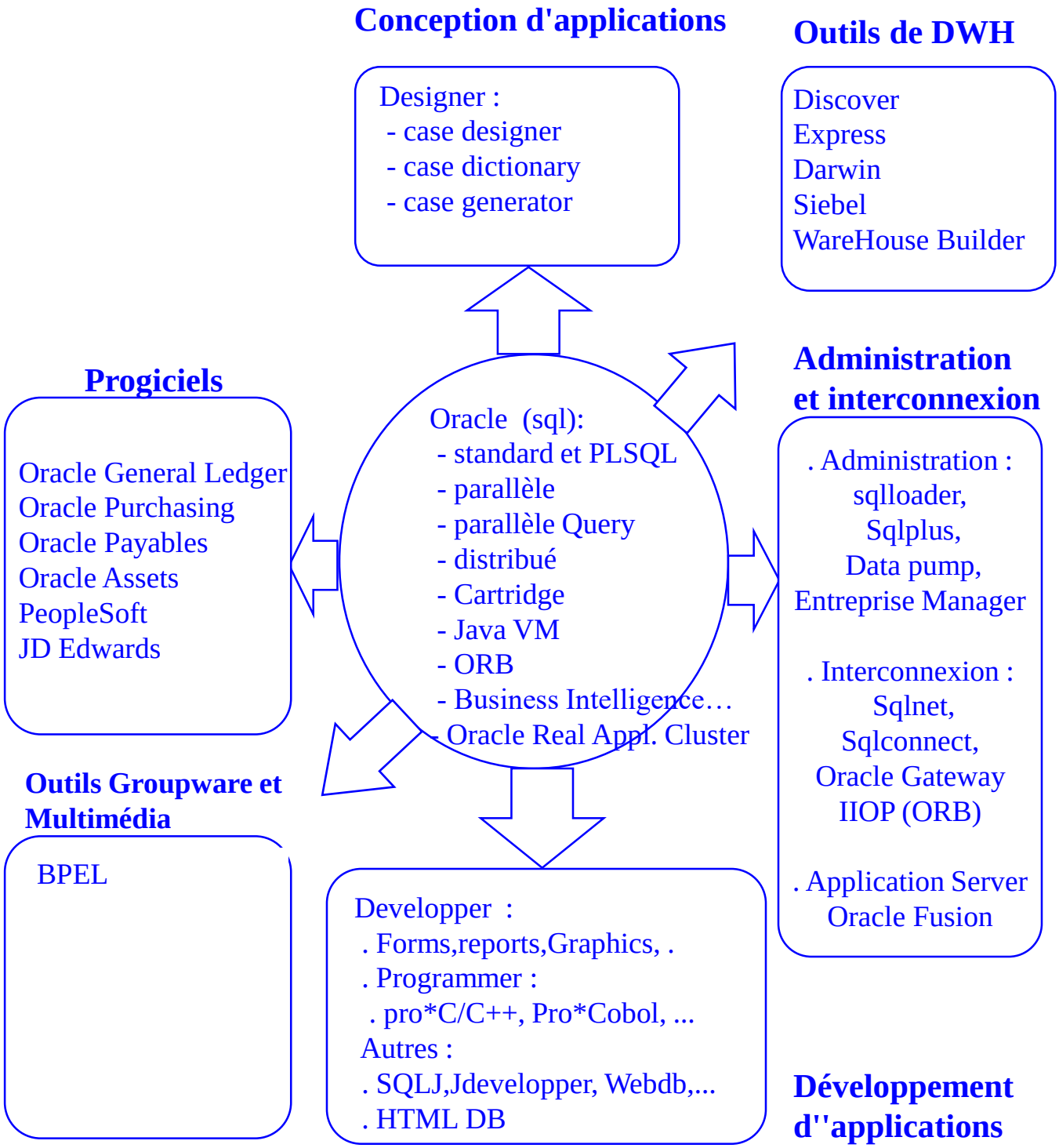
  - **Est un langage ensembliste (non procédural)**

  - **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

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.

1.4 Les objets

- Quelques objets du SGBD Relationnel ORACLE sont les suivants :

  - **Les Tables,**
  - **Les Vues,**
  - **Les Index,**
  - **Les Séquences,**
  - **Les Synonymes,**
  - **Les Clusters**
  - **Les procédures stockées**
  - **Les triggers**
  - **Les types abstraits**
  - **Les database links**
  - **Les utilisateurs**
  - **Les tablespaces**
  - **…**

- Seuls les objets **TABLES, VUES, INDEX, SYNONYMES et les SEQUENCES** seront étudiés dans ce cours.

2. Interrogation des données

2.1. Syntaxe du verbe SELECT

SELECT [ALL | DISTINCT] {[schéma.obj].*
| expr [c_alias], ...}
FROM [schéma].obj [t_alias], [[schéma].obj
[t_alias], ...]
[**WHERE** <condition>]
[**CONNECT BY** <condition>
[**START WITH** <condition>]]
[**GROUP BY** expr, expr, ...
[**HAVING** <condition>]]
[**ORDER BY** {expr|pos} [**ASC**|**DESC**],
[{expr|pos} [**ASC**|**DESC**], ...]

[] : Facultatif

{} : choix obligatoire

| : choix entre différentes options

2.1. Syntaxe du verbe SELECT

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***

  - ***obj*** : peut être une TABLE, une VUE ou une Vue matérialisée

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

  - ***c_alias*** est le renommage de l'expression

  - ***t_alias*** est le renommage d'une table, vue ou vue matérialisée

2.2. Indépendance logique externe

- Architecture ANSI X3/SPARC

Niveau Externe

PRIX	PROD#
1000	10

….
Redisposition des colonnes

Nr.	Cout	Prix Produit
10	Coûte	1000

….
* Renommage des colonnes
* Constantes

EMP#	Augmentation
1	12000

….
Calcul horitaux
Exemple : SAL*1.1

TOTAL SALAIRES
300000

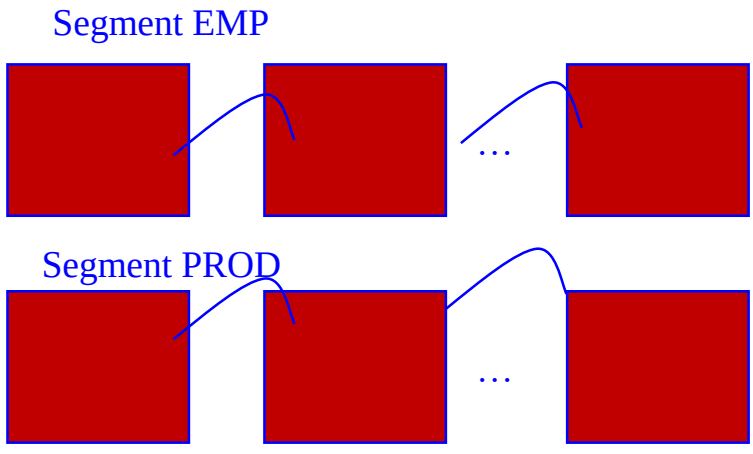
….
Calcul Verticaux
Exemple : SOMME(SAL)

Niveau logique

Table EMP (EMP#,NOM,ADR,SAL)

Table PROD(Prod#, PRIX)

Niveau Physique



2.2. Indépendance logique externe

Renommage des colonnes et des tables

- alias d'attributs et
- alias des tables, vues ou vues matérialisées

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

14 ligne(s) sélectionnée(s).

2.2. Indépendance logique externe

Alias des tables

- **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

14 ligne(s) sélectionnée(s).

2.2. Indépendance logique externe

Redisposition des colonnes

SQL> desc pilote;

Nom	Non renseigné	NULL?	Type
PL#	NOT NULL		NUMBER(4)
PLNOM	NOT NULL		VARCHAR2(12)
DNAISS	NOT NULL		DATE
ADR			VARCHAR2(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

2.2. Indépendance logique externe

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;
```

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

2.2. Indépendance logique externe

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 ANNUEL"
        FROM pilote;
```

PL#	SALAIRE ANNUEL
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

14 ligne(s) sélectionnée(s).

2.2. Indépendance logique externe

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 SUM(cap) "CAPACITE TOTALE"
        FROM avion;

CAPACITE TOTALE
 5170
```

2.3. Elimination de doublons : DISTINCT

- Le mot clé DISTINCT dans la clause SELECT
  - Réalise un tri sur les colonnes et
  - Elimine les doublons.

- **Exemple**

SQL> SELECT DISTINCT avtype FROM avion;

AVTYPE
A300
A320
B707
B727
Caravelle
Concorde

Il est possible de faire un DISTINCT de plusieurs colonnes.

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 Série 1

1.1 Alias des attributs

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

Numéro Nom Adresse Salaire Mensuel

1.2 Redisposition des attributs

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

Nom Salaire Mensuel Numéro Adresse

1.3 Alias d'une table

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

1.4 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.

1.5 Calculs verticaux

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

1.6 Distinct

Donner tous les types d'avions de la compagnie

2.4. Opération de sélection

La clause WHERE

SELECT ... FROM ...

WHERE[NOT] prédicat1 [AND|OR] [NOT] prédicat2...

- La clause WHERE permet d'effectuer un filtrage des lignes d'une table. C'est à dire sélectionner un sous-ensemble de lignes dans des tables.
- Seules les lignes vérifiant la clause WHERE seront retournées.
- Prédicat

  EXPRESSION1 **OPERATEUR** EXPRESSION2

  EXPRESSIONi peut être : un nom de colonne, une constante ou Expression

  OPERATEUR peut être : >,>=, <, <=, NOT =!,

- Notes
  - Les connecteurs 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

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 et SOUNDEX

- **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 et SOUNDEX

- Le mot clé ESCAPE permet de déspécialiser les caractères jokers :

  _
  et
  %.

- Le caractère précisé derrière le mot clé 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 '*';
```

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

Opérateur SOUNDEX

- SOUNDEX(chaîne) est une fonction qui permet une comparaison phonétique.
- SOUNDEX(chaîne) génère une valeur numérique sur 4 octets (selon un algorithme précis).
- Expression pour faire une comparaison phonétique entre 2 chaînes :

  SOUNDEX(chaîne1) = SOUNDEX(chaîne2)

Exemple :

Sélectionnez les pilotes dont le nom ressemble à Tonton

```
SQL> SELECT plnom
        FROM pilote
     WHERE SOUNDEX(plnom) =
SOUNDEX('Tonton');
```

PLNOM

Tintin

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

Opérateur SOUNDEX

- Algorithme de l'opérateur SOUNDEX

1. Garder la 1ère lettre de la chaîne et supprimer toutes les occurrences des lettres suivants : a, e, h, i, o, u, w, y.

TINTIN =>Tntn

TONTON=>Tntn

2. Affecter des nombres aux lettres restantes après la première comme suit :

b, f, p, v = 1

c, g, j, k, q, s, x, z = 2

d, t = 3

l = 4

m, n = 5

r = 6

Tntn=>T535

3. Si deux ou plusieurs lettres avec le même nombre sont adjacente dans le nom original (avant l'étape 1), ou adjacent excepté pour les h et les w alors les omettre tous sauf le premier

4. Retourner les premiers 4 bytes complétés de 0

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.

- **Exemple**

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, pour pas de valeur définie).

```
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.
- 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

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 Série 2

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

2.2 "Nom des pilotes habitants Nice ou Paris"

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

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

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

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

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

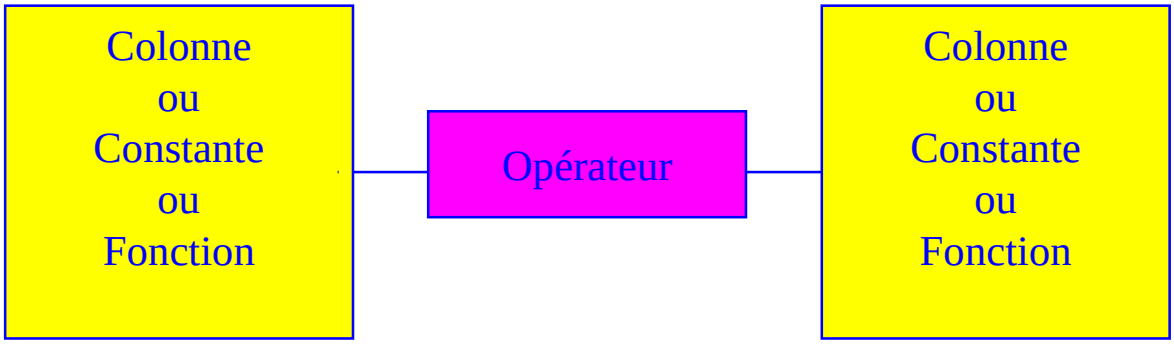
2.8 "Nom des pilotes ayant 2 'o' dans leur nom "

2.9 "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



- **Opérateur** peut être
  - Opérateurs arithmétiques : + - * /
  - Opérateur sur chaînes de caractères : ||

- **Exemple 1**

  Simuler une augmentation de 10% des salaires des pilotes

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

- **Exemple 2**

  Pilotes dont le salaire est supérieur à 2 fois 10000

```
SQL> SELECT *
        FROM pilote
        WHERE sal > 10 000 * 2;
```

2.5.1. Les expressions

- **Exemple 3**

  Ajouter 3 jours à une date

  '08-DEC-90' + 3 = '11-DEC-90'

- **Exemple 4**

  Enlever 3 jours à une date

  '11-DEC-90' - 3 = '08-DEC-90'

- **Exemple 5**

  Nombre de jours entre 2 dates

  date1 - date 2 = nbjours

2.5.1. Les expressions

- **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).
```

2.5.2. Les fonctions

Fonctions sur les numériques

- **ABS(n)** : valeur absolue de n

  Exemple :

  ABS(-125) = 125

- **SIGN(n)** : signe de n (-1 ou 0 ou +1)

  Exemple :

  SIGN(-125) = -1
  SIGN(125) = 1
  SIGN(0) = 0

- **CEIL(n)** : plus petit entier >= n

  Exemple :

  CEIL(-125) = -125
  CEIL(125.3) = 126
  CEIL(125.7) = 126

2.5.2. Les fonctions

Fonctions sur les numériques

- **FLOOR(n)** : plus grand entier <= n

  Exemple :

  FLOOR(-125) = -125
  FLOOR(125.3) = 125
  FLOOR(125.5) = 125
  FLOOR(125.7)= 125

- **MOD(m,n)** : reste de la division de m par n

  Exemple :

  MOD(-125, 7) = -6
  MOD(125.3, 7.2) = 2.9
  MOD(125, 7) = 6

- **POWER(m, n)** : m élevé à la puissance n

  Exemple :

  POWER(-10, 7) = -10000000

- **SQRT(n)** : racine carrée de n (Erreur si n < 0)

  Exemple :

  SQRT(100) = 10

2.5.2. Les fonctions

Fonctions sur les numériques

- **ROUND(n,[m])** : arrondi de n à 10-m

  Ex emple :

  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

Fonctions sur les chaînes de 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 lg.

  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

Fonctions sur les chaînes de caractères

- **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'

2.5.2. Les fonctions

Fonctions sur les chaînes de caractères

- **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

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

  Exemples :

  REPLACE('TUTU','U', 'OU') = 'TOUTOU'

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

- **TRANSLATE(chaîne,car_source,car_cible) :** transcodage de certains caractères par d'autres caractères dans une chaîne de caractères.

  Exemples :

  TRANSLATE('ARMAND','AM','12')='1R21ND'

  - le A est remplacé par 1
  - le M est remplacé par 2

  dans les noms des pilotes

2.5.2. Les fonctions

Fonctions sur les chaînes de caractères

- **SOUNDEX(chaîne) : (voir 2.7.2)**

- **ASCII(chaîne) :** donne la correspondance ASCII du premier caractère de la chaîne.
  Exemple : **ASCII('ADFGRSE') = 65**

- **CHR(n) :** caractère correspondant à la valeur de n en ASCII.
  - **Exemple :** **CHR(65) = 'A'**

2.5.2. Les fonctions

Fonctions de manipulation de dates

- **LAST_DAY(date)** : Date du dernier jour du mois d'une date donnée

  Exemple :

  LAST_DAY ('10-12-1987') = '31/12/87'

- **NEXT_DAY(date, jour)** : date du prochain jour à partir d'une date donnée.

  Exemple :

  NEXT_DAY ('10-12-1987', 5) = '14/12/87'

- **ADD_MONTHS(date,n)** : ajoute n mois à une date donnée.

  Exemple :

  ADD_MONTHS ('10-12-2007', 2) = '10/02/2008'

- **MONTHS_BETWEEN(date1,date2)** : nombre de mois entre 2 dates.

  Exemples :

  MONTHS_BETWEEN ('10/02/2008', '10-12-2007')=2

2.5.2. Les fonctions

Fonctions de manipulation de dates

- **ROUND(date[,'precision'])** : arrondi d'une date en fonction de la précision

Exemples : SYSDATE = '12-JUL-96'

ROUND(sysdate,'MM') = '01-JUL-96'

ROUND(sysdate + 4 ,'MM') = '01-AUG-96'

ROUND(sysdate,'YY') = '01-JAN-97'

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

Exemples

TRUNC(sysdate,'MM') = '01-JUL-96'

TRUNC(sysdate + 4 ,'MM') = '01-JUL-96'

TRUNC(sysdate,'YY') = '01-JAN-96'

2.5.2. Les fonctions

Format de date pour les fonctions ROUND et TRUNC

Format	Unité d'arrondi et de troncature
CC	Le plus grand pas rapport aux deux premiers chiffres d'une année en 4 chiffre
SCC	
SYYYY	Année (arrondir au premier Janvier de l'année suivante si la date courante commence au 1er Juillet. Sinon arrondir au premier janvier de l'année courante)
YYYY	
YEAR	
SYEAR	
YYY	
YY	
Y	
IYYY	Année ISO
IY	
IY	
I	

2.5.2. Les fonctions

Format de date pour les fonctions ROUND et TRUNC

Exemple 1

ROUND(to_date('17-02-**20**07'), ‘CC') = ‘01-01-20**01**‘

ROUND(to_date('15-02-2007') , 'SYYYY') ='01-01-2007'
ROUND(to_date('17-07-2007') , 'SYYYY') ='01-01-2008'

ROUND(to_date('15-02-2007') , ‘IYYY') ='01-01-2007'
ROUND(to_date('17-07-2007') , ‘IYYY') ='31-12-2007'

2.5.2. Les fonctions

Format de date pour les fonctions ROUND et TRUNC

Format	Unité d'arrondi et de troncature
Q	Quarter (arrondi vers le 1er jour du trimestre suivant à partir du 16ème jour du 2ème mois du trimestre courant. Sinon arrondi vers le 1er jour du trimestre courant)
MONTH	Month (Arrondi au 1er jour du mois suivant à partir du 16ème jour du mois courant. Sinon arrondi vers le 1er jour du mois courant)
MON	
MM	
RM	
WW	Même jour de la semaine que le premier jour de l'année. Si le premier jour de l'année c'est un lundi, revenir au lundi précédent la date actuelle avant le 16ème. Aller au lundi suivant à partir du 16ème jour
IW	Même jour de la semaine que le premier jour de l'année ISO
W	Même jour de la semaine que le premier jour du mois

2.5.2. Les fonctions

Format de date pour les fonctions ROUND et TRUNC

Exemple 2

ROUND(to_date('17-02-2007'), 'Q') = ‘01-04-2007‘

ROUND(to_date('15-02-2007'), 'Q') = ‘01-01-2007‘

ROUND(to_date('17-02-2007'), ' MONTH') = '01-03-2007‘

ROUND(to_date('15-02-2007'), ' MONTH ') = ‘01-02-2007‘

Le premier jour de 2007 est un Lundi.

ROUND(to_date('15-02-2007'), '), ' WW') =’12/02/2007’

ROUND(to_date('17-02-2007'), '), ' WW') = ’19/02/2007’

2.5.2. Les fonctions

Format de date pour les fonctions ROUND et TRUNC

Format	Unité d'arrondi et de troncature
DDD	Jour
DD	
J	
DAY	Arrondi au Jour de démarrage de la semaine courante avant le 16ème jour du mois et au premier jour de la semaine suivante après le 15ème jour du mois.
DY	
D	
HH	Heure
HH12	
HH24	
MI	Minute

2.5.2. Les fonctions

Format de date pour les fonctions ROUND et TRUNC

Exemple 3

ROUND(to_date('17-02-2007'), ‘DDD') = ‘15-02-2007‘
ROUND(to_date('15-02-2007'), ‘DDD') = ‘17-02-2007‘

ROUND(to_date('17-02-2007'), ‘DAY') = ‘12-02-2007‘
ROUND(to_date('15-02-2007'), ‘DY') = ‘19-02-2007‘

```
select to_char(sysdate, 'DD-MM-YYYY :
    HH24')"Date_Sans_Arrondi_Heure" FROM DUAL;
```

Date_Sans_Arrondi_Heure

15-01-2008 : 10

```
select to_char(ROUND(sysdate, 'HH'), 'DD-MM-YYYY :
    HH24') "Date_Avec_Arrondi_Heure" FROM DUAL;
```

Date_Avec_Arrondi_Heure

15-01-2008 : 11

2.5.2. Les fonctions

Fonctions de conversion

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

  Exemple : TO_NUMBER('567') = 567

- **TO_CHAR(val[,'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

Fonctions de conversion

• 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;
```

==> 01 07 1996 10 : 24

2.5.2. Les fonctions

Fonctions de conversion

- 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 :

```
Select TO_CHAR(SYSDATE,' "LE" DAY DD MONTH
    YYYY "A" HH24 : MI')
From dual;
```

==> LE VENDREDI 03 FÉVRIER 2012 A 08 : 46

2.5.2. Les fonctions

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.

  Exemple :

  Select plnom, decode(tel, null, 'Pas de tél.', 'Tél : '|| tel) "Info Tél" From Pilote;

PLNOM	Info Tel
Miranda	Tel : 93548254
St-exupery	Tel : 91548254
…	
Vernes	Pas de tel.
Tournesol	Pas de tel.
…	

EXERCICES Série 3

3.1 "Lister les pilotes avec leur salaire tronqués au millier"

3.2 "Lister les pilotes avec leur salaire. Pour ceux gagnant 17000,6

remplacer le salaire par '****' "

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

3.4 Calculer la durée d'un vol de façon plus exacte

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 maximum**
  - **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.
  - Liste non exhaustive

- **Exemple**
  - SQL> SELECT AVG(sal), COUNT(sal), MAX(sal), MIN(sal), STDDEV(sal),SUM(sal), VARIANCE(sal) from pilote;

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.

- **La clause ORDER BY**

ORDER [SIBLINGS] BY {expr | position | c_alias}
[ASC | DESC] [NULL FIRST | NULL LAST]

Option	Description
SIBLINGS	valide uniquement si requête hiérarchique. Préserve l'ordre de tri hiérarchique
Exp	Expression de tri : un nom de colonne par exemple
Position	Position de l'expression (colonne) dans la clause SELECT
C_alias	Alias d'une expression (colonne)
ASC	Tri ascendant (par défaut)
DESC	Tri descendant

NULL FIRST Faire apparaître les Null en premier

NULL LAST Faire apparaître les null en dernier (par défaut)

2.7. Présentation du résultat trié selon un ordre précis

Remarques

- Par défaut en Oracle le tri est toujours ascendant.
- Dans un tri par ordre croissant les valeurs NULL apparaissent toujours en dernier sauf indication contraire
- Les **colonnes** peuvent être désignées par leur **nom ou leur numéro.**

Exemple

```
-- Trier en ordre ascendant selon le nom
SQL> SELECT plnom, adr FROM pilote
          ORDER BY plnom;

-- Trier en ordre ascendant sur adr et descendant sur
   plnom
SQL> SELECT plnom, adr FROM pilote
          ORDER BY 2 , plnom desc;
```

2.8. Utilisation des pseudo colonnes

Généralités

- A chaque table Oracle est associée **un ensemble de colonnes implicites.** Ces colonnes sont aussi appelées **Pseudo Colonnes**. Citons par exemple les colonnes ROWID, USER, SYSDATE, ROWNUM, …

- **ROWID**
  - L'adresse d'un tuple, composé de quatre champs:
    - Le numéro de bloc dans le fichier,
    - Le numéro de tuple dans le bloc et
    - Le numéro de fichier,
    - Le numéro de segment

- **USER :** L'utilisateur actuellement connecté

- **SYSDATE :** la date système

- **ROWNUM :** Numéro des lignes résultats d'une requête SQL

2.8. Utilisation des pseudo colonnes

- **Exemple**

Select Sysdate, user From Pilote;

SYSDATE	USER
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM
08-JUN-99	SYSTEM

…

16 rows selected.

La valeur de Sysdate et User est repétée autant de fois qu'il ya de lignes dans la table.

2.8. Utilisation des pseudo colonnes

La table DUAL

- **La table DUAL 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
08-JUN-99	SYSTEM

EXERCICES Série 4

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

<valeur à calculer> "Max salaire Pilote "

4.2 "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 ? "

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

4.4 "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

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

Généralités

- L'objectif **de la jointure est de ramener sur une même ligne le résultat des informations venant de différentes tables**.
- Décomposition de la jointure :
  1. **Sélection (restriction)**
  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 s'il y a ambigüité**.
- 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 sans sous requêtes : la jointure ou produit cartésien

Exemple 1 : sans auto jointure

Donner la liste des pilotes qui assurent un vol au départ de Nice.

```
Select distinct plnom
From Pilote p, Vol v
Where p.pl#=v.pilote# and vd = 'Nice';
```

PLNOM

Armstrong
Bush
Chretien
Gagarine
Miranda
Mopolo
Tintin

7 rows selected.

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

Exemple 2 : avec auto jointure

Donner les couples des Pilotes qui gagnent le même Salaire.

```
Select p1.plnom, p2.plnom, p1.sal
From Pilote p1, Pilote p2
Where p1.sal=p2.sal and p1.plnom >p2.plnom
Order by sal;
```

PLNOM	PLNOM	SAL
Tournesol	Chretien	15000.6
Mopolo	Icare	17000.6
Vernes	Icare	17000.6
Vernes	Mopolo	17000.6
Ruskoi	Bush	22000

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

Exemple 3 : Jointure interne

Donner les numéros de vols ainsi que les noms des pilotes, les types d'avions qui participent à ces vols.

```
Select  v1.vol#,  p1.plnom, a1.avtype, pl#, pilote#, av#,
    avion#
From Pilote p1, Avion a1, vol v1
Where p1.pl#=v1.pilote# and a1.av#=v1.avion# ;
```

VOL#	PLNOM	AVTYPE	PL#	PILOTE#	AV#	AVION#
290	Armstrong	B727	3	3	8	8
110	Armstrong	B707	3	3	6	6
280	Bush	CONCORDE	8	8	9	9
135	Bush	CONCORDE	8	8	5	5
236	Bush	A300	8	8	4	4
140	Chretien	CONCORDE	14	14	9	9
210	Chretien	CARAVELLE	14	14	7	7
310	Foudelle	B727	19	19	8	8
111	Gagarine	A320	5	5	3	3
200	Gagarine	A320	5	5	3	3

2.10. Requêtes multi-relations avec les opérateurs ensemblistes

Généralités

- **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

Règles

- Même nombre de variables en projection
- Correspondance de 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)

2.10. Requêtes multi-relations avec les opérateurs ensemblistes

Exemple

- Considérons une compagnie aérienne dont les pilotes sont gérés par pays. Il peut être intéressant pour la direction d'avoir accès à l'ensemble des pilotes de l'entreprise.

```
Select plnom, sal from pilote_france
UNION
Select plnom, sal from pilote_usa
UNION
Select plnom, sal from pilote_Allemagne ;
```

Restrictions

- Ne s'applique pas sur les colonnes de type : LONG,LONG ROW, BLOB, CLOB, BFILE, VARRAY, NESTED TABLE
- Si une expression apparaît dans la liste de projection un alias est nécessaire si l'on souhaite trier sur cette colonne
- Impossible d'utiliser la clause FOR UPDATE
- Impossible de trier chaque sous-requête

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 opérateurs ALL, IN, ANY, SOME.

- **Remarque**

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

2.11. Sous-interrogations non synchronisée

Exemple lié au cas 1

- Donner la liste des Pilote dont le salaire est supérieur à celui du pilote N°1.

Select plnom, sal from pilote

Where sal > **(Select sal From Pilote Where pl# =1);**

PLNOM	SAL
Armstrong	24500
Tintin	21100
Gagarine	22100
Baudry	21000
Bush	22000
Ruskoi	22000
Yen	29000
Concorde	1000.6

8 rows selected.

2.11. Sous-interrogations non synchronisée

Exemple lié au cas 2

- Donner la liste des Pilote dont le salaire est supérieur à celui des pilote qui habitent Paris.

```
Select plnom, sal from pilote
Where sal >ALL (Select sal From Pilote
                Where adr= 'Paris');
```

PLNOM	SAL
Tintin	21100
Ruskoi	22000
Bush	22000
Gagarine	22100
Armstrong	24500
Yen	29000

2.11. Sous-interrogations non synchronisée

L'opérateur EXISTS

- La sous-interrogation ramène VRAI s'il existe au moins une ligne en réponse à la sous-interrogation, FAUX sinon.
- **La requête principale s'exécute si VRAI.**

Syntaxe :

```
SELECT ...
FROM ...
WHERE [NOT] EXISTS (SELECT ...)
```

- **Exemple**

Donner la liste des Pilotes qui gagnent plus que tous les pilotes habitant Paris.

```
Select plnom, sal, adr From Pilote p1
WHERE Not Exists(Select p2.sal From Pilote p2
 Where p2.adr = 'Paris' and p1.sal<=p2.sal);
```

PLNOM	SAL	ADR
Armstrong	24500	Wapakoneta
Tintin	21100	Bruxelles
Gagarine	22100	Klouchino
Bush	22000	Milton
Ruskoi	22000	Moscou
Yen	29000	Munich

6 rows selected.

2.12. Sous-interrogations synchronisée

Principe

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

  SELECT ...

  FROM ...

  WHERE variable Op ?

  (voir plus haut)

- Mais lorsque **la valeur ?** est susceptible de varier pour chaque ligne de la REQUETE PRINCIPALE, on utilise les sous-interrogations synchronisées.

Règles

- Le prédicat de la sous-interrogation fait référence à une colonne de la requête principale

- Si une table est présente dans les 2 select, la renommer

Ordre d'exéxcution

- L'exécution de la sous-interrogation se fait pour chaque ligne de l'interrogation principale.

2.12. Sous-interrogations synchronisée

- ***Exemple***

Donner la liste des vols des Pilotes en service qui gagnent entre 20000 et 22000.

```
Select distinct vol#, pilote# From Vol v
Where pilote# in
(Select p.pl# From Pilote p
    Where v.pilote#=p.pl# And
     p.Sal>20000 and p.Sal<22000);
```

VOL#	PILOTE#
120	4
130	4

2.13. La jointure externe

- La jointure externe ("outer join") **se comporte comme l'opérateur de jointure avec en plus la possibilité de ramener des informations sur les lignes n'ayant pas vérifier la condition de jointure**.

- Cette fonctionnalité est directement offerte par SQL d'Oracle, en faisant suivre, *dans la condition de jointure, la colonne de la table qui ne contient pas l'information recherchée, par le signe "(+)"*.

- L'opérateur **"(+)"** nous permettra par exemple en un seul trait de lister les noms des Pilotes qui conduisent avec leurs Vols et pour ceux qui ne conduisent pas uniquement leurs noms.

2.13. La jointure externe

Contraintes

- L'opérateur de jointure externe "(+)" doit **apparaître au plus une fois dans un même prédicat de jointure**
- Pas de prédicat avec IN ou OR admis
- **Si plusieurs prédicats de jointures existent entre deux tables A et B l'opérateur (+) doit apparaître dans chacun des prédicat**. En cas de non respect Oracle traite uniquement comme une jointure interne.
- Pas de outer join sur un prédicat de jointure si une table est utilisée comme étant OUTER et l'autre comme étant INNER
- L'opérateur (+) ne s'applique pas si une seule table est référencée derrière la clause FROM
- L'opérateur (+) ne peut apparaître que derrière la clause WHERE

2.13. La jointure externe

Exemple :

- Donner la liste Pilotes et les numéros de vol auxquels ils participent. Pour ceux qui ne participent à aucun, leur nom doit être affiché.

```
Select plnom, vol# From Pilote p, Vol v
Where p.pl# = v.pilote#(+);
```

PLNOM	VOL#
Miranda	100
Miranda	150
St-exupery	153
Armstrong	110
Armstrong	290
Tintin	120
Tintin	130
Gagarine	111
Gagarine	200
Baudry	
Bush	135
Bush	280
Bush	236
Ruskoi	156
Math	
Yen	
Icare	125
…	

2.13. La jointure externe

Jointure externe avec l'utilisation du mot clé OUTER JOIN

- La jointure externe avec l'opérateur (+) est spécifique à Oracle. Elle comporte plusieurs limites citées précédemment

- Il est conseillé d'utiliser à la place OUTER JOIN qui ne comporte pas de limitation et est de surcroit normalisé

- **Syntaxe**

… FROM Table_reference
{FULL | LEFT | RIGHT} [OUTER] JOIN
Table_reference ON join_condition

Nota : Left désigne la table à gauche et Right la table à droite. Indique la table pour laquelle on veut ramener les lignes qui ne vérifient pas la condition de jointure

2.13. La jointure externe

Jointure externe avec l'utilisation du mot clé OUTER JOIN

• Exemple 1 : LEFT OUTER JOIN

– Donner la liste Pilotes et les numéros de vol auxquels ils participent. Pour ceux qui ne participent à aucun, leur nom doit être affiché.

```
Select plnom, vol# From Pilote p left outer join Vol v
on p.pl# = v.pilote# ;
Select plnom, vol# From Vol v  right outer join Pilote p
on p.pl# = v.pilote# ;
```

Est équivalent à

```
Select plnom, vol# From Pilote p, Vol v
Where v.pilote#(+) =p.pl# ;
```

• Exemple 2 : RIGHT OUTER JOIN

– Donner la liste Pilotes et les numéros de vol auxquels ils participent. Afficher aussi les VOLs sans pilote

```
Select plnom, vol# From Pilote p right outer join Vol v
on p.pl# = v.pilote# ;
```

Est équivalent à

```
Select plnom, vol# From Pilote p, Vol v
Where v.pilote#= p.pl# (+) ;
```

2.13. La jointure externe

Jointure externe avec l'utilisation du mot clé OUTER JOIN

- **Exemple 3 : FULL OUTER JOIN**

  – Donner la liste Pilotes et les numéros de vol auxquels ils participent. Afficher aussi les VOLs sans pilote. Afficher aussi les pilotes qui n'assurent aucun VOL

```
Select plnom, vol# From Pilote p FULL  outer join Vol v
on p.pl# = v.pilote# ;
```

Est équivalent

```
Select plnom, vol# From Pilote p, Vol v
Where p.pl# = v.pilote# (+)
Union all
Select plnom, vol# From Pilote p, Vol v
Where p.pl# (+)= v.pilote# ;
```

EXERCICES Série 5

5.1 Requêtes avec alias obligatoires (auto-jointure) et préfixage d'attributs(naming)

5.1.1 "Donner toutes les paires de noms de pilotes distincts, habitant la même ville"

5.2 Requêtes effectuant une jointure syntaxique

5.2.1 "Donner tous les noms des pilotes qui ont des noms d'avions ?"

5.2.2 "Ecrire la requête qui donne les noms des pilotes qui conduisent un A300 ou B727 ?".

5.2.3 "Tester la requête suivante :

```
(SELECT PILOTE#, VD, VA
FROM vol)
        INTERSECT
(SELECT AVION#, VD, VA
FROM VOL
);
```

Quel est sa signification en langage naturel ?

EXERCICES Série 5

5.3 ***Sous-requêtes connectées par les opérateurs ANY, ALL, EXISTS, IN.***

5.3.1 "Quel est le nom des avions dont la capacité est supérieure à la capacité de chaque avion localisé à Nice ?"

5.3.2 "Quel est le nom des avions dont la capacité est au moins égale à celle d'un avion localisé à Nice ? (jamais > ou <) "

5.3.3 "Quel est le nom des pilotes assurant un vol au départ de Nice ?"

5.3.4 "Quel est le nom des pilotes assurant au moins un vol ?"

5.4 ***Requêtes multi-relations avec sous-requêtes indépendantes ou jointure externe***

5.4.1 "Quel est le nom des pilotes dont le salaire est supérieure au salaire maximum de tous les pilotes effectuant un vol au départ de Paris ?"

5.4.2 "Quels sont les noms des pilotes qui gagnent plus que le pilote nr. 5?"

5.4.3 "Donner le nom des pilotes, et pour ceux qui sont en service, la liste des numéros de vols qu'ils assurent ?"

2.14. La recherche hiérarchique

Généralités

- Le langage SQL d'Oracle permet la représentation et la manipulation de données ayant une structure hiérarchique.

- Les lignes résultats sont ordonnées selon le parcours de l'arbre.

- Le niveau de la hiérarchie dans lequel se trouvent les données concernées par la recherche peut-être accessible par la pseudo-colonne (attribut implicite de la table), LEVEL, jusqu'à 256 niveaux.

```
SELECT   <colonne [, colonne, ...] >
FROM schema.table
[WHERE   <expr>]
CONNECT BY [NOCYCLE] PRIOR <colonne 1>=<colonne2>
    [AND <condition>]
    [START WITH <Condition>] ;
```

- **Notes** :
  - **START WITH** : ligne(s) de départ de construction de l'arborescence (racine de l'arbre)
  - **CONNECT BY** fixe le parcours de la hiérarchie (lien père-fils)
  - **PRIOR** : colonne de départ
  - **NOCYCLE** : évite les boucles

2.14. La recherche hiérarchique

- **Exemple :**

"Quels sont les vols en correspondance (direct ou indirecte) au départ de Paris ?"

Note : NICE ne doit pas être une escale de départ.

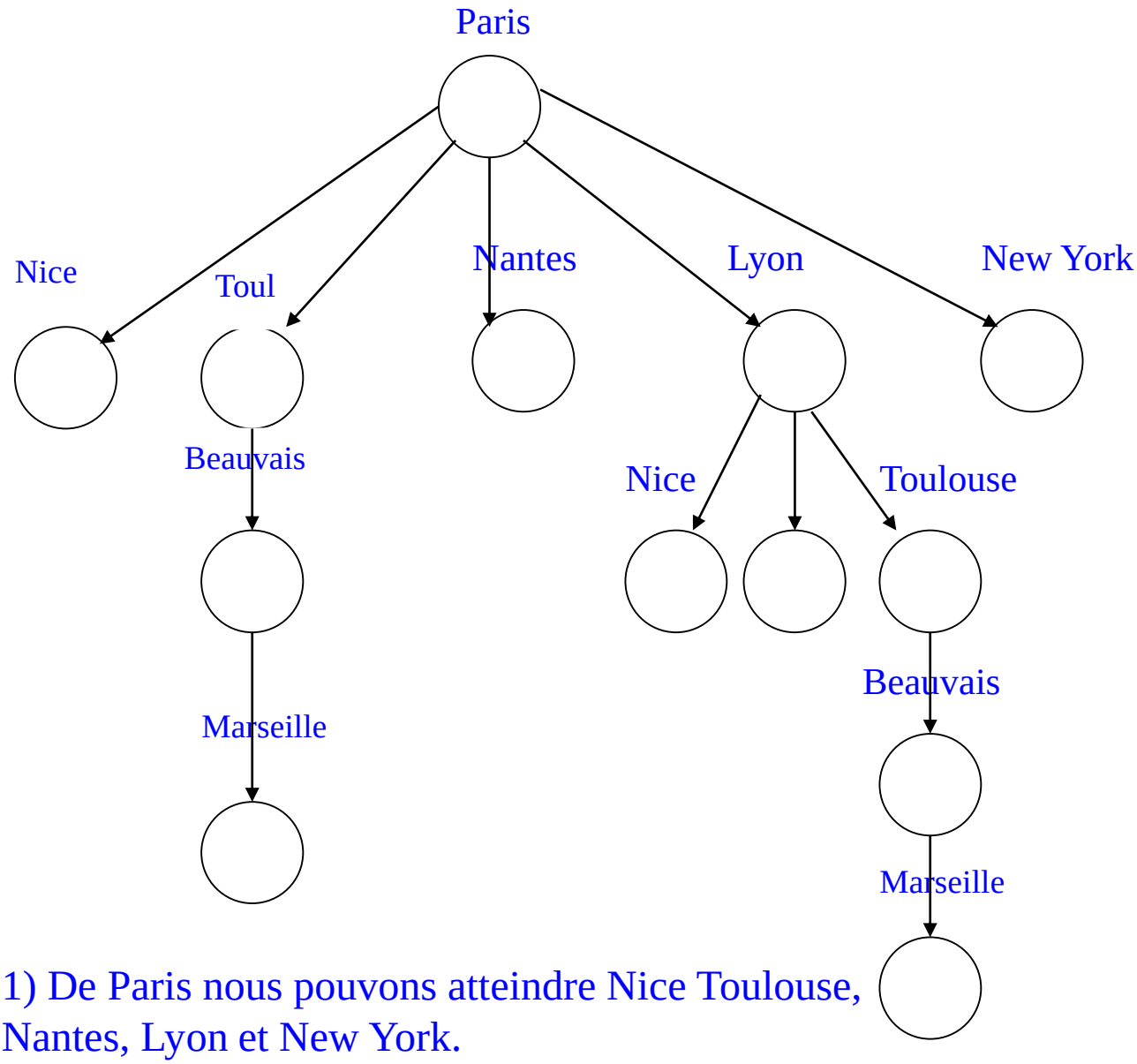
```
SQL> column vol_en_corres format A30;
SQL> Select LPAD(' ',2*(LEVEL),'  ')
       || vol# vol_en_corres, vd, va
    From vol
       START WITH vd='Paris'
       CONNECT BY PRIOR va=vd AND vd != 'Nice';
```

Vol_en_Corres	VD	VA
125	Paris	Nice
135	Paris	Toulouse
130	Toulouse	Beauvais
290	Beauvais	Marseille
150	Paris	Nantes
156	Paris	Lyon
140	Lyon	Nice
153	Lyon	Nice
236	Lyon	Toulouse
130	Toulouse	Beauvais
290	Beauvais	Marseille
270	Paris	New york

12 **rows selected.**

2.14. La recherche hiérarchique

Arborescence de l'exemple précédent



1) De Paris nous pouvons atteindre Nice Toulouse, Nantes, Lyon et New York.
2) La branche Nice n'est pas développée car Nice ne doit pas être une ville de Départ.
3) De Toulouse nous pouvons atteindre Beauvais. …

2.14. La recherche hiérarchique

Etapes de traitement

- **1. Calcul des enregistrements racines**

  Ce sont les lignes qui satisfont le prédicat de la clause START WITH

- **2. Calcul des lignes filles pour chaque Enregistrement Racine**

  Chaque ligne fille doit satisfaire la condition de la clause CONNECT BY.

- **3. Calcul récursif des lignes filles** pour chaque ligne fille de l'étape au dessus. Répéter autant que nécessaire la logique de l'étape 2

- **4. En cas de présence d'une clause Where, les lignes ne vérifiant pas la clause ne participe pas à la construction de la hiérarchie**

- **5. Restitution de l'arborescence**

2.14. La recherche hiérarchique

Remarque sur le prédicat de sélection

- S'il est placé à l'extérieur de la clause CONNECT BY, il élimine certaines valeurs uniquement (le parcours de l'arborescence n'est pas interrompu)

- Si il est placé à l'intérieur de la clause CONNECT BY, il élimine certaines valeurs et leurs dépendances (le parcours de l'arborescence est interrompu).

- Limites

  – Une requête hiérarchique ne doit pas contenir de jointure

  – Une requête hiérarchique ne peut être effectuée sur une vue de jointure

  – La clause ORDER BY est prioritaire à la clause CONNECT BY

2.14. La recherche hiérarchique

Détection des boucles

- ORACLE détecte les boucles éventuelles dans le CONNECT BY. L'ordre est interrompu et un message d'erreur est envoyé à l'utilisateur.

- **Exemple**

```
Select LPAD(' ',2*(LEVEL),'  ')
       || vol# vol_en_corres, vd, va
   From vol
    START WITH vd='Paris'
    CONNECT BY PRIOR va=vd ;
```

ERROR:
ORA-01436: CONNECT BY loop in user data

2.14. La recherche hiérarchique

Détection des boucles

- Il est possible d'utiliser le mot clé NOCYCLE pour éviter les boucles

- **Exemple**

```
Select LPAD(' ',2*(LEVEL),'  ')
      || vol# vol_en_corres, vd, va
   From vol
    START WITH vd='Paris'
    CONNECT BY NOCYCLE PRIOR va=vd;
```

VOL_EN_CORRES	VD	VA
125	Paris	Nice
100	Nice	Paris
135	Paris	Toulouse
130	Toulouse	Beauvais
320	Beauvais	Marseille
290	Beauvais	Marseille
310	Beauvais	Marseille
150	Paris	Nantes
156	Paris	Lyon
236	Lyon	Toulouse

2.14. La recherche hiérarchique

La Pseudo Colonne LEVEL

Cette colonne permet de contrôler le Niveau d'une arborescence et d'éviter aussi des boucles.

- **Exemple**

```
Select LPAD(' ',2*(LEVEL),'  ')
      || vol# vol_en_corres, vd, va
   From vol
      START WITH vd='Paris'
      CONNECT BY PRIOR va=vd and LEVEL<3;
```

VOL_EN_CORRES	VD	VA
125	Paris	Nice
100	Nice	Paris
110	Nice	Toulouse
120	Nice	Paris
111	Nice	Paris
200	Nice	Toulouse
210	Nice	Nantes
240	Nice	Paris
280	Nice	Mulhouse
135	Paris	Toulouse
130	Toulouse	Beauvais

…

17 rows selected

2.14. La recherche hiérarchique

Les Pseudo Colonnes CONNECT_BY_ISCYCLE et CONNECT_BY_ISLEAF et la fonction SYS_CONNECT_BY_PATH

- **CONNECT_BY_ISCYCLE** : renvoie 1 si Enregistrement E1 père de E2 est aussi fils de de E2. 0 sinon. S'utilise uniquement avec NOCYCLE
- **CONNECT_BY_ISLEAF** : renvoie 1 si un enregistrement n'a pas de fils 0 sinon
- **SYS_CONNECT_BY_PATH** : Cette fonction renvoie le chemin d'une hiérarchie par rapport à un père

2.14. La recherche hiérarchique

La Pseudo Colonne CONNECT_BY_ISCYCLE et CONNECT_BY_ISLEAF

- **Exemple**

```
Select vol#, vd, va,
CONNECT_BY_ISLEAF "feuille",
CONNECT_BY_ISCYCLE "Cycle",
LEVEL, SYS_CONNECT_BY_PATH(va, '/') "chemin"
From vol
    START WITH vd='Paris'
    CONNECT BY NOCYCLE PRIOR va=vd;
```

VOL#	VD	VA	feuille	Cycle	LEVEL	chemin
125	Paris	Nice	0	0	1	/Nice
100	Nice	Paris	1	1	2	/Nice/Paris
135	Paris	Toulouse	0	0	3	/Nice/Paris/Toulouse
130	Toulouse	Beauvais	0	0	4	/Nice/Paris/Toulouse/Beauvais
320	Beauvais	Marseille	1	0	5	/Nice/Paris/Toulouse/Beauvais/Marseille
290	Beauvais	Marseille	1	0	5	/Nice/Paris/Toulouse/Beauvais/Marseille
310	Beauvais	Marseille	1	0	5	/Nice/Paris/Toulouse/Beauvais/Marseille
150	Paris	Nantes	1	0	3	/Nice/Paris/Nantes
156	Paris	Lyon	1	1	3	/Nice/Paris/Lyon
236	Lyon	Toulouse	0	0	4	/Nice/Paris/Lyon/Toulouse

EXERCICES Série 6

6.1 "Quels sont les vols en correspondance (direct ou indirecte) au départ de Paris ?"

Note :

NICE ne doit pas être une escale de départ.

6.2 « Afficher la hiérarchie des employés dans une l'entreprise en partant du PDG dont le nom est KING »

Note : Vous devez pour cela installer le script demobld.sql pour avoir la table EMP table des employés.

2.15. Le partitionnement

Généralités

- Le partitionnement permet de regrouper les lignes résultat en fonction des différentes valeurs prises par une colonne spécifiée derrière la clause GROUP BY.

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

- La spécification de la clause GROUP BY entraîne la création d'autant de sous-groupes 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 HAVING joue le rôle de filtre pour la clause GROUP BY**.
L'exécution de la clause HAVING sera effectuée juste après celle du GROUP BY, pour sélectionner les sous-groupes qui satisfont la condition spécifiée.

2.15. Le partitionnement

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.**

Exemple

Quel est la capacité moyenne des avions par ville et par type.

```
Select loc, avtype, AVG(cap) "Cap Moyenne"
From avion  Group by loc, avtype;
```

LOC	AVTYPE	Cap Moyenne
Nice	A300	300
Nice	CONCORDE	300
Paris	A300	350
Paris	A320	320
Paris	B707	400
Paris	CARAVELLE	300
Toulouse	B727	250
Toulouse	CONCORDE	350

8 rows selected.

2.15. Le partitionnement

Elimination de groupe via HAVING

- Exemple

  Quel est la capacité moyenne des avions par ville et par type. Garder les groupes qui ont au moins deux avions de même type.

```
Select loc, avtype, AVG(cap) "Cap Moyenne"
From avion
--Where loc =
Group by loc, avtype
Having count(avtype) >1
;
```

LOC	AVTYPE	Cap Moyenne
Nice	A300	300
Paris	A300	350

EXERCICES Série 7

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

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

7.3 "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.**

- **Ordres SQL de mises à jour des données dans une base Oracle :**

  - Insertion (INSERT),
  - Suppression (DELETE)
  - Modification (UPDATE)

3.1. Insertion de lignes

- **Ordre Insert**

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

- ***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

Ordre UPDATE

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

Ordre DELETE

DELETE FROM <[nom_user].nom_table | nom_vue>
[WHERE <critères_de_qualification>];

- **Exemple :**

Supprimer tous les vols au départ de Nice

```
SQL> DELETE FROM VOL
            WHERE vd='Nice';
```

Supprimer tous les vols

```
SQL> DELETE FROM VOL ;
```

3.3. Suppression de lignes

Via la commande TRUNCATE

TRUNCATE TABLE nom_table
[DROP STORAGE | REUSE STORAGE]

- 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

3.3. Suppression de lignes

Via la commande DROP TABLE

DROP TABLE nom_table [cascade constraints]

- Cette commande permet d'effectuer la suppression de la table et de les objets dépendants (indexes etc.). C'est une commande du LDD d'Oracle et à ce titre équivaut à un commit.

- **Exemple :**

  SQL> DROP TABLE pilote;

- ***Remarque :***
  - La table, les indexes, les contraintes d'intégrés sont supprimés

3.4. Avantages / Inconvénients des 3 solutions

Suppression via 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.

Suppression via DROP TABLE

- Tous les index, contraintes d'intégrité et triggers associés à la table sont également supprimés
- Tous les GRANT sur cette table sont supprimés

3.4. Avantages / Inconvénients des 3 solutions

3ème option TRUNCATE :

- **Truncate est plus rapide** que DELETE 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 Série 8

8.1 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.

8.2 Effectuer une insertion dans la table PILOTE2 via une sous-requête sur PILOTE.

8.3 Mettre à jour le salaire du pilote numéro 3 à 19000 F et Valider.

8.4 Supprimer le pilote numéro 11 et invalider.

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

4. Le schéma de données

4.1 Les principaux objets d'une base Oracle

Généralités : Les principaux objets d'une base Oracle

- **La table** : unité de base pour le stockage des données dans une base oracle.
- **La vue** : sous ensemble logique d'une ou plusieurs tables et/ou autres vues
- **La séquence** : servent à générer automatiquement les clés
- **L'index** : accélérateur de requêtes
- **Les synonymes** : autre nom d'un objet
- **Le database link** : lien vers le schéma d'un utilisateur distant
- **La vue matérialisée**: copy asynchrone d'une table distante
- **Le trigger** : alerte ou déclencheur (permet d'introduire la notion d'événement dans une base de données)
- **Procédures et packages** : procédures stockées
- **Le cluster** : jointure physique entre deux ou plusieurs tables.
- **Type** : type défini par l'utilisateur (l'option objet)
- …

- **Note :**
  - Dans ce cours nous ne traitons que les objets : table, index, vue, séquence et synonyme.

4.2 Les règles de nommage des objets d'une base

Règles

- Le *nom d'une objet* doit commencer par une lettre
- Sa *longueur* peut être de 1 à 30 caractères
- Ces noms *ne doivent contenir que des lettres, des chiffres, "_", "$" et "#".*
- Un *même objet dans le même schéma* ne peut avoir le même nom. Le nom exacte d'un objet est
  - Schema.objet Schéma : Utilisateur propriétaire et objet : nom de l'objet
  - Ou schema.objet@DBLINK : dblink est le pointeur vers une base distante
- Le nom d'un objet ne peut *être un mot clé ou mot réservé* d'Oracle

4.3 Les types de données

Les principaux types de données Oracle

- **CHAR(taille)** : Chaîne - longueur fixe - de 1 à 255 octets
- **VARCHAR2(taille)** : Chaîne de taille variable 1...2000 bytes
- **VARCHAR(taille)** : Idem varchar2 (type réservé pour les versions futures d'Oracle : ne pas utiliser)
- **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

4.3 Les types de données

Les principaux types de données Oracle

- **CLOB** : nouveau type texte (jusqu'à 4Go)
- **BLOB** : nouveau type binaire (jusqu'à 4 Go)
- **BFILE** : données binaires stockées dans un fichier externe
- ***Remarques*** :
  - 1) une seule colonne de type LONG ou LONG RAW par table. Ces limites sont rompus avec Oracle et ses Larges Object LOB.
  - 2) l'utilisateur **peut maintenant créer ses propres types** grâce à la commande CREATE TYPE
  - 3) dans une même table, plusieurs colonnes de type CLOB, BLOB et BFILE peuvent être définies.

4.4 Comparaison entre chaînes

- **Comparaison varchar2 / char**

- **Comparaison 1:**
  - Oracle ajoute des blancs au char le plus petit pour que les 2 chaînes aient la même longueur.
  - Oracle compare ensuite car. par car. Dès qu'il y a une différence il conclut lequel est plus grand.

  ==> utilisée si les 2 valeurs à comparer sont de type CHAR

- **Comparaison 2 :**
  - Oracle compare car. par car. Dès qu'il y a une différence il conclut lequel est plus grand. Si on a atteint la fin du 1er varchar sans rencontrer de différence avec le deuxième, c'est le plus long (le 2ème) qui est considéré comme le plus grand.

  ==> utilisée dès qu'une des 2 valeurs est de type varchar2 dans l'expression.

COMP1	COMP2
'ab' > 'aa'	**'ab' > 'aa'**
'ab' > 'a '	**'ab' > 'a '**
'ab' > 'a'	**'ab' > 'a'**
'ab' = 'ab'	**'ab' = 'ab'**
'a ' = 'a'	**'a ' > 'a'**

4.5 Création d'une table

Généralités

- une table est l'unité de base pour le stockage des données dans une base oracle

- Deux types de tables subsistent sous Oracle 8. Ce sont les tables dites objets et les tables relationnelles classiques. *Nous étudions ici uniquement les tables relationnelles classiques*

- Une table peut être créée à partir d'une requête

Syntaxe

```
CREATE TABLE [user.]<nom_table>
    [(
     colonne          typeDeDonnée   [DEFAULT expr]
                      [contrainteAuNiveauColonne],
     … ,
     [contrainteAuNiveauTable]

    )]
| as subquery};
```

4.5 Création d'une table

Syntaxe

- Avec :
  - **User** : nom du propriétaire de la table
  - **Table** : nom de la table
  - **Default expr** : valeur par défaut d'une colonne en cas d'omission lors de l'insertion
  - **Colonne** : nom de la colonne
  - **TypeDeDonnée** : le type d'une colonne
  - **ContrainteAuNiveauColonne** : spécification d'une contrainte d'intégrité au niveau colonne
  - **ContrainteAuNiveauTable** : spécification d'une contrainte d'intégrité au niveau table
  - **SubQuery** : Création d'une table à partir d'une sous-requête

4.5 Création d'une table

Exemple

- **Exemple 1 :** Création d'une table sans contraintes d'intégrités.

```
create table pilote(
      pl#      number(4)    ,
      plnom    varchar2(12),
      dnaiss   date         ,
      adr      varchar2(20)      ,
      tel      varchar2(12),
      sal      number(7,2)
     );
```

- **Exemple 2 : Création dune table à partir d'une requête**

```
Create table Pilote2
     As Select * from Pilote;
```

La table Pilote2 sera peuplée des lignes de la table Pilote.

4.6 Les contraintes d'intégrité

Généralités

- Les contraintes d'intégrité permettent de vérifier l'intégrité des données au niveau du schéma

- La gestion des contraintes au niveau du schéma allège d'autant le code des applicatifs

- Les contraintes d'intégrité nous évitent de supprimer malencontreusement les données dépendantes (intégrité de référence)

- Les principaux types de contraintes valables sous Oracle sont :
  - **NOT NULL** : refus de valeurs nulles pour une colonne
  - **UNIQUE** : indique que les valeurs d'une ou plusieurs colonnes doivent être unique
  - **PRIMARY KEY** : identificateur unique d'une ligne
  - c'est aussi la contrainte d'intégrité d'entité
  - **FOREIGN KEY/REFERENCES** : permet de gérer l'intégrité de référence
  - **CHECK** : permet de gérer l'intégité de domaines.

4.6 Les contraintes d'intégrité

Généralités

- Il est **conseillé de nommer ses contraintes** sinon Oracle affecte un nom par défaut ayant le format suivant SYS_Cn. Exemple SYS_C0001;

- Les contraintes peuvent être définies au niveau colonne ou au niveau table..

4.6 Les contraintes d'intégrité

Définition d'une contrainte au niveau Colonne

- une contrainte est définie au niveau colonne lorsqu'elle n'implique pas des informations d'autres colonnes

- **Syntaxe**

colonne typeCol [CONSTRAINT nomContrainte]
typeContrainte ;

Définition d'une contrainte au niveau table

- ***une contrainte est définie au niveau table lorsqu'elle implique plus d'une colonne***
- Lorsqu'elle est définie après la création d'une table grâce à la commande Alter Table

- **Syntaxe**

CREATE TABLE nomTable (…
,
[CONSTRAINT nomContrainte]
typeContrainte (colonne, …), …

Note

- NomContrainte : nom de la contrainte
- TypeContrainte : type de la contrainte

4.6 Les contraintes d'intégrité

La contrainte PRIMARY KEY

- 1. Permet de définir un identifiant unique des lignes d'une table
- 2. Peut être définie sur une ou plusieurs colonne
- 3. un index unique est créé implicitement par Oracle pour assurer l'unicité

- **Exemple** 1: définition au niveau colonne

```
Create table Pilote(
Pl#        Number(4) Constraint pk_pilote_pl#
                primary key,…
);
```

- **Exemple** 2: définition au niveau Table

```
Create table LigneCommande(
Commande#       Number(4),
LigneCommande# number (4),
…,
Constraint pk_LigneCommande Primary
key(Commande#, LigneCommande#)
);
```

4.6 Les contraintes d'intégrité

PRIMARY KEY vs UNIQUE key

	P.K.	Unique
1. toutes les valeurs sont distinctes	oui	oui
2. la colonne est définie en NOT NULL	oui	pas oblig.
3. définit l'identifiant des lignes	oui	
4. précisé une seule fois par table	oui	
5. fait le lien avec REFERENCES	oui	

4.6 Les contraintes d'intégrité

La contrainte FOREIGN KEY

- 1. permet de définir une clé étrangère pour assurer l'intégrité de référence
- 2. les valeurs de la clé étrangère doivent correspondre à celle de la clé primaire référencée
- 3. les mots clés permettant de définir les clés étrangères sont : FOREIGN KEY, REFERENCES et ON DELETE CASCADE
- 4. le mot clé REFERENCE utilisé permet de définir des contraintes d'intégrité de référence au niveau colonne
- 5. l'option **ON DELETE CASCADE** permet de propager les suppressions au lignes dépendantes (lignes contenant les FOREIGN KEY)

4.6 Les contraintes d'intégrité

La contrainte FOREIGN KEY

- 6. l'option **ON DELETE SET NULL** permet d'initialiser les colonnes FOREIGN KEY à NULL en cas de suppression de la clé primaire dépendante

- 7. le mot clé **FOREIGN KEY** associé à **REFERENCES** permet de **définir une contrainte d'intégrité de référence au niveau table** ;

- 8. une FOREIGN KEY peut être définie sur une ou plusieurs colonnes.

4.6 Les contraintes d'intégrité

La contrainte FOREIGN KEY

- ***Exemple 1***

Niveau colonne :

```
CREATE TABLE VOL(
…,
Pilote#  number(4) constraint fk_vol_pilote#
REFERENCES Pilote(pl#),
);
```

Niveau Table :

```
CREATE TABLE VOL  (
…,
Pilote#  number(4),
constraint fk_vol_pilote#  FOREIGN KEY (pilote#)
      REFERENCES Pilote(pl#)
);
```

4.6 Les contraintes d'intégrité

La contrainte FOREIGN KEY

- ***Exemple 2***

Supprimer aussi les lignes dépendantes automatiquement si la ligne père est supprimée

```
CREATE TABLE VOL(
…,
Pilote#  number(4) constraint fk_vol_pilote#
REFERENCES Pilote(pl#) ON DELETE CASCADE,
);
```

Mettre les colonnes FOREIGN KEY à null automatiquement si la ligne père est supprimée

```
CREATE TABLE VOL  (
Pilote#  number(4),
…,
constraint fk_vol_pilote#  FOREIGN KEY (pilote#)
      REFERENCES Pilote(pl#)
            ON DELETE SET NULL
);
```

4.6 Les contraintes d'intégrité

La contrainte CHECK

- 1. permet d'assurer l'intégrité de domaine
- 2. permet de définir une condition que doit vérifier chaque ligne
- 3. Certaines expressions ne sont pas autorisées par exemple :
  - Faire référence aux pseudo colonnes CURRVAL, NEXTVAL, LEVEL et ROWNUM
  - Appeler les fonctions SYSDATE, UID, USER et USERENV
  - introduire des requêtes qui se réfèrent au valeurs d'autres lignes

- **Exemple**

Niveau colonne :

```
CREATE TABLE PILOTE(…
sal        number(7,2) not null
                    CHECK(sal < 70000.0));
```

Niveau table :

```
 CREATE TABLE VOL(…
      ha        number(4),
      hd        number(4)  NOT NULL,
      …,
 CONSTRAINT ck_ha CHECK(HA > HD));
```

4.6 Les contraintes d'intégrité

- **Exemple général**

```
create table pilote(
    pl#       number(4) constraint pk_pilote_pl#  primary key,
    plnom     varchar2(12)   constraint nl_pilote_plnom not
    null constraint uk_pilote_plnom unique,
    dnaiss    date           constraint nl_pilote_dnaiss not null,
    adr       varchar2(20)     default 'PARIS',
    tel       varchar2(12),
    sal       number(7,2)      not null check (sal < 70000.0)
);

create table avion(
    av#       number(4)        not null primary key,
    avtype    varchar2(10)     not null
     CONSTRAINT avion_chk_type
     CHECK (avtype in
    ('A300','A310','A320','B707','B727','CONCORDE',
    'CARAVELLE')),
    cap   number(4)   not null,
    loc       varchar2(20)     not null,
    remarq    long
);
```

4.6 Les contraintes d'intégrité

Exemple général

```
create table vol(
    vol#     number(4)      primary key,
    pilote# number(4)       not null
    CONSTRAINT vol_fk_pilote REFERENCES
    PILOTE(PL#)
    ON DELETE CASCADE,
    avion#      number(4)      not null,
    vd       varchar2(20),
    va       varchar2(20),
    hd       number(4)  not null,
    ha       number(4),
    dat      date,
    CONSTRAINT vol_chk_ha CHECK (ha>hd),
    FOREIGN KEY (avion#) REFERENCES
    AVION(AV#)
);
```

4.7 Création d'un index

Généralités

- Les index permettent d'accéder plus rapidement aux données.
- Ils servent également à gérer l'unicité des clés primaires ou unique : un index UNIQUE est créé implicitement sur la ou les colonnes identifiant la clé primaire.
- Les index sont stockés dans une structure externe à la table.
- On peut créer plusieurs index sur une table.
- Les index sont mis à jour par ORACLE lors des ordres INSERT, UPDATE, DELETE.

Syntaxe

```
CREATE [UNIQUE] INDEX  nom_index
ON nom_table(colonne, colonne, ...) ;
```

- Paramètres et mots clés

Unique	**:** index unique
Nom_index	**:** nom de l'index à créer
Nom_table	**:** nom de la table
Colonne	**:** la ou les colonnes à indexer

4.7 Création d'un index

Exemples

- **Exemple 1 : index non unique**

  Create index idx_sal_pilote ON pilote (sal);

- **Exemple 2 : index unique**

  Create Unique index idx_plnom_pilote
  ON pilote (plnom);

- **Exemple 3 : index concaténé**

  Create index idx_plnom_sal_pilote
  ON pilote (plnom, sal);

EXERCICES Série 9

EXERCICES Série 9

9.1 "Créer une relation FORMATION, qui contiendra les renseignements suivants :

- le numéro de pilote ,

- le type de formation (ATT, VDN, PAI, ...)

- type d'appareil

- date de la formation "

Attention : - un pilote à une date donnée participe à une formation

- un type d'appareil doit être : 'A300', 'A310', 'A320', 'B707', 'Caravelle', 'B727'

ou 'Concorde'

La clé primaire doit être définie sur le numéro du pilote le type de formation et la date de formation.

9.2 Créer un index unique sur la colonne PLNOM de PILOTE. Que constatez vous.

9.3 Créer également un index sur la colonne AVTYPE de la table FORMATION.

9.4 insérer deux lignes dans la table Formation (2 formations pour le pilote Nr 3)

4.8 Modification d'une table

- **La modification d'une table est possible à travers la clause ALTER TABLE.**
  - ALTER TABLE [<nom_user>.] <Table> ...

- **La clause ALTER TABLE permet :**

  - 1. **d'ajouter de nouvelles colonnes**

    ALTER TABLE [<nom_user>.] <Table>
    ADD <def_col> ...

    – **Exemple**

      Alter table Pilote
      Add (age number(3)) ;

  - 2. **d'ajouter de nouvelles contraintes d'intégrité**

    ALTER TABLE [<nom_user>.] <Table>
    ADD <table_contrainte> ...

    – **Exemple**

      Alter table Pilote
      Add (constraint chk_pilote_age Check (age Between 25 AND 70));

4.8 Modification d'une table

La clause ALTER TABLE permet :

- **3. de redéfinir une colonne** (type de données, taille, valeur par défaut)

ALTER TABLE [<nom_user>.] <Table>
MODIFY <def_col> ...
avec <def_col> :
(colonne[typeDeDonnées] [DEFAULTexpr]
contrainteAuNiveauColonne]) ;

– **Exemple**

```
Alter Table Pilote
        Modify (sal number(7,2));

Alter Table Pilote
        Modify (sal number(7,2) not null);

Alter Table Pilote
        Modify (adr   Default  'Nice');
```

4.8 Modification d'une table

La clause ALTER TABLE permet :

- *Note :*
  - **NOT NULL est la seule contrainte pouvant être ajoutée par MODIFY. Des valeurs nulles ne doivent pas déjà avoir été stockées**
  - **La taille d'une colonne ne peut être réduite si des données existent dans la table.**

- **4. de modifier les paramètres de stockages (voir cours administration)**

- **5. d'allouer explicitement des extensions dans des fichiers précis(voir cours d'admnistration)**

- **6. De supprimer des colonnes**

```
Alter table Pilote
DROP COLUMN age ;
```

4.8 Modification d'une table

La clause ALTER TABLE permet :

- **7. d'activer/désactiver/supprimer une contrainte d'intégrité**

ALTER TABLE [<nom_user>.] <Table>
ENABLE <clause> | DISABLE <clause> |
DROP <clause> ...

avec <clause> :
UNIQUE (col1[,col2 ...]) [CASCADE] |
PRIMARY KEY
[CASCADE] |
CONSTRAINT <nom_contrainte> [CASCADE]

Paramètres et mots clés :
ENABLE : active une contrainte
DISABLE : désactive une contrainte
DROP : supprime une contrainte
CASCADE: désactive les contraintes dépendantes

4.8 Modification d'une table

La clause ALTER TABLE permet :

- **8. d'activer/désactiver/supprimer une contrainte d'intégrité**

  – ***Exemple***

  1. Désactivation d'une contrainte

  Alter Table Pilote Disable
  Constraint pk_pilote_pl# cascade ;

  2. Activation d'une contrainte

  Alter Table Pilote enable
  Constraint pk_pilote_pl# ;

  3. Suppression d'une contrainte

  Alter Table Pilote Drop
  Constraint pk_pilote_pl#;

4.8 Modification d'une table

Restrictions aux modifications des tables

- 1. AJOUT
  - On peut ajouter une colonne de type NOT NULL uniquement si la table est vide
  - On peut ajouter une contrainte uniquement au niveau table

- 2. MODIFICATION
  - On peut retrécir une colonne uniquement si elle est vide
  - On peut passer une colonne de NULL autorisé à NOT NULL uniquement si la colonne ne contient pas de valeur NULL
  - On ne peut modifier une contrainte

- 3. SUPPRESSION
  - On peut supprimer une colonne
  - On peut supprimer une contrainte par son nom

4.9 Définition des commentaires

Commentaires sur les tables ou les colonnes

- Le commentaire sur une colonne se fait par la clause SQL suivante :

  COMMENT ON
  {TABLE nom_table |
  COLUMN table.colonne} IS chaîne ;

Note

- Les commentaires sont insérés dans le Dictionnaire de Données.Leur consultation se fait entre autre à travers la vue USER_COL_COMMENTS.

Exemple :

```
SQL> COMMENT ON TABLE Pilote IS  'Pilotes de la
     compagnie';

SQL> COMMENT ON COLUMN pilote.pl#
          IS 'Numéro identifiant le pilote';
```

Pour supprimer un commentaire :

```
SQL> COMMENT ON COLUMN pilote.pl#
          IS '';
```

410 Consultation de la structure d'une table

- **Clause de listage des colonnes d'une table :**

  DESC[RIBE] [user.]nom_table ;

- **La clause DESCRIBE permet de lister les colonnes d'une table. L'utilisateur doit être propriétaire de la table ou en avoir reçu les droits.**

- **Exemple:**

SQL> DESC pilote;

Name	Null?	Type
PL#	NOT NULL	NUMBER(4)
PLNOM	NOT NULL	VARCHAR2(12)
DNAISS	NOT NULL	DATE
ADR		VARCHAR2(20)
TEL		VARCHAR2(12)
SAL	NOT NULL	NUMBER(9,2)

4.11 Création d'un synonyme

Synonyme d'une table

- Un synonyme est utilisé pour la sécurité et la facilité de manipulation.

Syntaxe

CREATE [PUBLIC] SYNONYM

[<user>.]<nomSynonyme>

FOR [<user>.]<nomTable> ;

Paramètres et mots clés

NomSynonyme : nom du synonyme

NomTable : nom de la table à rebaptiser.

Public : le synonyme est accessible par tous les users.

ATTENTION : son utilisation abusive augmente le temps d'exécution des requêtes.

Notes sur le synonyme

- Sert à référencer les objets sans indiquer leur propriétaire
- Sert à référencer les objets sans indiquer leur base
- Fournit un autre nom à un objet : alias
- Un synonyme privé doit avoir un nom distinct dans le schéma d'un utilisateur
- Un synonyme public peut avoir le nom de la table dans son schéma.

4.11 Création d'un synonyme

Exemple

```
CREATE PUBLIC SYNONYM Vol
 FOR scott.vol;
```

Remarque

- On peut également créer des synonymes pour des vues, séquences, procédures, ... et même des synonymes.

4.12 Les séquences

Généralités

- 1. une séquence est un générateur de nombre unique
- 2. permet de générer des clés primaires
- 3. diminue le coût de codage des applications

Création d'une séquence

- Syntaxe

```
CREATE SEQUENCE sequence
    [INCREMENT BY n]
    [START WITH n]
    [{MAXVALUE n | NOMAXVALUE}]
    [{MINVALUE n | NOMINVALUE}]
    [{CYCLE | NOCYCLE}]
    [{CACHE n | NOCACHE}]
```

Paramètres et mots clés

Sequence : nom de la séquence

INCREMENT BY n : incrémenter de n en n

START WITH n : démarrer à partir de n

4.12 Les séquences

Paramètres et mots clés

- **MAXVALUE n** : spécifie la valeur maximale d'une séquence
- **NOMAXVALUE** : valeur maximale 10^27 pour une séquence ascendante et –1 pour une séquence descendante
-
- **MINVALUE n** : spécifie la valeur minimale d'une séquence
- **NOMINVALUE** : valeur minimale de 1 pour une séquence ascendante et -(10^26) pour une séquence descendante
- **CYCLE :** continue à générer les valeurs même si le min ou le max sont attteints
- **NOCYCLE** : plus de génération de valeur si min ou max sont atteints
- **CACHE n**
- NOCACHE : spécifie le nombre de valeurs à préallouer

4.12 Les séquences

Exemple de création d'une séquence

```
Create Sequence seq_pilote_pl#
INCREMENT BY 1
START WITH 1
MAXVALUE 2000
CACHE 10
NOCYCLE;
```

NEXTVAL et CURRVAL

- Nextval : rend la prochaine valeur disponible de la séquence
- Currval : rend la valeur courante de la séquence

Utilisation des Séquences

```
-- insertion dans une table
INSERT INTO Pilote
Values (seq_pilote_pl#.nextval, 'Bill', …);
-- consultation de la valeur courante
Select seq_pilote_pl#.currval From dual;
Insert into vol
Values(600, seq_pilote_pl#.currval , …);
```

EXERCICES Série 10

10.1 "Ajouter la colonne AGE à la table PILOTE. Un pilote doit avoir entre 25 et 60 ans.

10.2 "Ajouter une contrainte d'intégrité de référence au niveau table à la relation FORMATION (colonne PILOTE#)"

10.3 "Modifier la colonne PL# de la table PILOTE en number(5).

10.4 Ajouter une valeur par défaut à la colonne VD dans VOL.

10.5 "Associer à l'attribut SAL (salaire) d'un pilote un commentaire puis s'assurer de son existence. Comment supprime - t-on un commentaire ?"

10.6 "Consulter la liste des colonnes de la table FORMATION"

10.7 "Attribuer un synonyme "Conducteurs" à la table PILOTE.

4.13 Le dictionnaire de données d'Oracle

Généralités

- Chaque base de données Oracle possède un dictionnaire de données
- Il répertorie tous les objets de la base et leur définition
- Le dictionnaire de données est un ensemble de tables dans lesquelles sont stockées les informations sur les schémas des utilisateurs.
- Le propriétaire des tables systèmes sous Oracle s'appelle SYS
- Le dictionnaire de données est mis à jour dynamiquement par ORACLE via le langage de définition de données LDD (CREATE, DROP, ALTER, …)

4.13 Le dictionnaire de données d'Oracle

Généralités

- Un des avantages des bases de données relationnelles est que l'accès aux informations du dictionnaire se fait à travers la clause SELECT-FROM-WHERE.

- Pour faciliter la consultation via SQL, il existe des vues et des synonymes systèmes

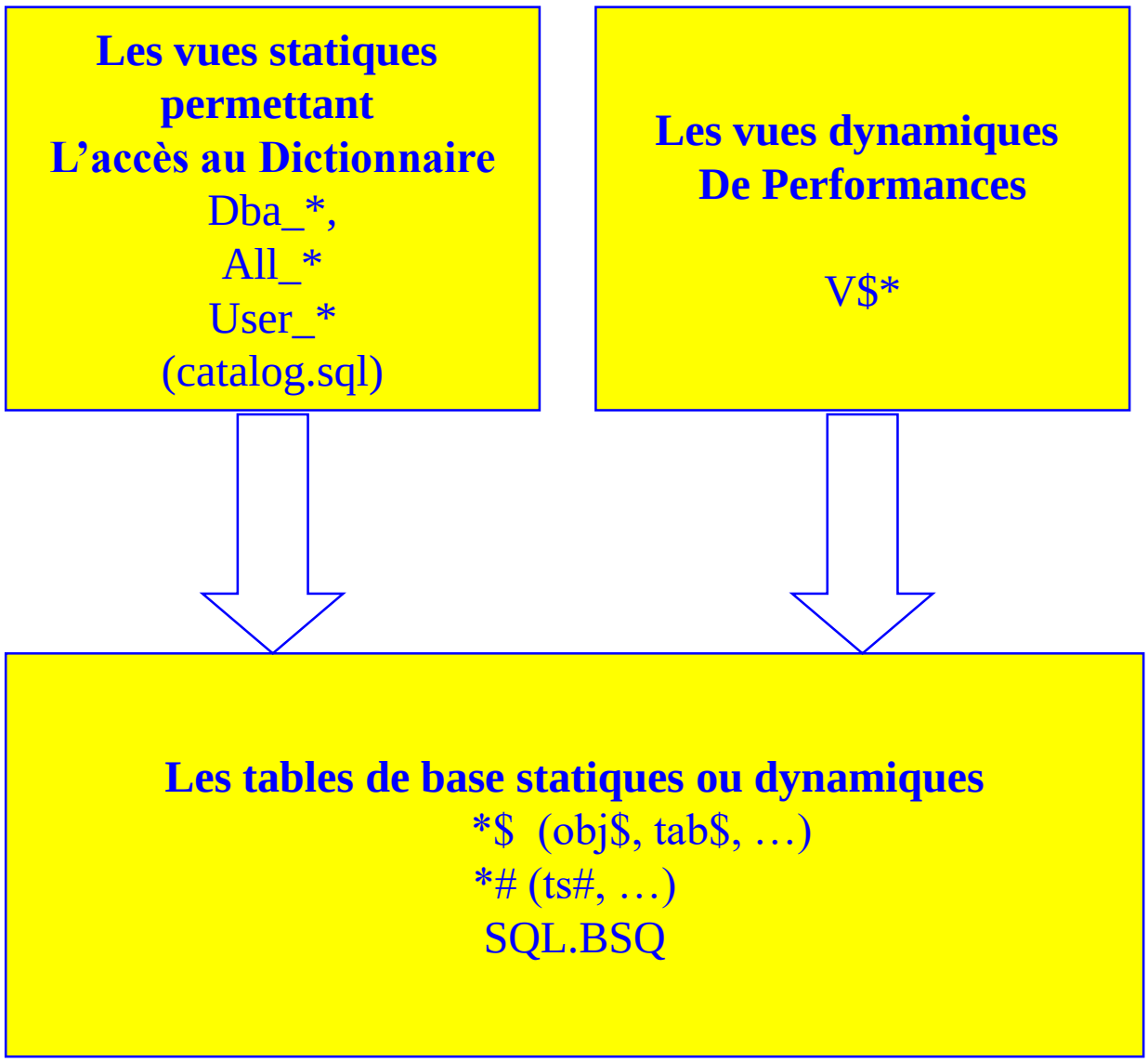
- ATTENTION, l'utilisateur n'accèdera aux informations que sur ses objets ou ceux sur lesquels il à les GRANTS nécessaires.

- NOTA : Les objets dans le dictionnaire d'Oracle **sont toujours stockés en majuscules**

4.13 Le dictionnaire de données d'Oracle

Généralités

- Structure du dictionnaire de données Oracle (L'accès au dictionnaire à travers les vues)



4.13 Le dictionnaire de données d'Oracle

Les tables de bases

- seg$: segments définis dans la base de données
- obj$: objets définis sur la base de données
- tab$: tables définies dans la base de données y compris les clusters
- ind$: index définis dans la base de données
- col$: colonnes définies dans la base de données
- ts$: tablespaces définis dans la base de données
- ...
- **Notes**
  - Tables accessibles uniquement par l'utilisateur SYS.
  - Tables se terminant par un $
  - Il est interdit de mettre à jour directement ces tables.

4.13 Le dictionnaire de données d'Oracle

Les vues statiques

- accessible_tables : contient les tables et vues accessibles par l'utilisateur
- all_catalog : tables, vues , synonym, ... accessibles par le user
- all_tab_columns: synonyme de la table *accessible_table*
- all_tab_grants : synonyme de la table table_privileges
- all_tables : description des tables accessibles par un user
- all_users: informations sur l'ensemble des users d'une base
- all_views: textes des vues accessibles par un utilisateur
- - dba_catalog: toutes les tables, les vues, synonymes et séquences de la base

4.13 Le dictionnaire de données d'Oracle

Les vues statiques

- dba_data_files: informations sur les fichiers de la base de données
- dba_free_space : espace restant libre dans les fichiers de la base
- dba_users : informations sur l'ensemble des users de la base
- dba_tables: informations sur l'ensembles des tables de la base
- dba_views: texte de toutes les vues de la base
- …

4.13 Le dictionnaire de données d'Oracle

Les vues statiques

- user_catalog : tables, vues, ... dont l'utilisateur est propriétaire
- user_free_space : Extent libre dans un tablespace accessible par le user
- user_indexes : Descriptions des index de l'utilisateur
- user_tables : tables dont l'utilisateur est propriétaire
- user_views : textes des vues de l'utilisateur
- user_users : info sur le user courant
- - ...

4.13 Le dictionnaire de données d'Oracle

Les vues statiques

- **Notes**
  - USER_* sont des vues donnant des informations sur les objets dont l'utilisateur est propriétaire
  - ALL_* sont des vues donnant des informations sur les objets auxquels ***l'utilisateur a accès***
  - DBA_* sont des vues sur tous les objets de la base
  - Les vues commençant par dba_ sont accessibles par le DBA
  - Les vues commençant par all_ et user_ sont accessibles le DBA et l'utilisateur.

4.13 Le dictionnaire de données d'Oracle

Les synonymes

- Alias de certaines vues statiques

- Exemple
  - cat : synonyme de la vue user_catalog
  - clu : synonyme de la vue user_clusters
  - cols : synonyme de la vue user_tab_columns
  - dict : synonyme de la vue DICTIONARY
  - ind : synonyme de la vue user_indexes
  - seq : synonyme de la vue user_sequences
  - syn : synonyme de la vue user_synonyms
  - tab : synonyme de la vue user_tables
  - ...

4.13 Le dictionnaire de données d'Oracle

Les tables dynamiques ou de performances

- v$process : informations sur les processus en cours
- v$bgprocess: descriptions des processus d'arrière plan
- v$licence: informations sur la validité des licenses
- v$lock : informations sur les verrous et les ressources
- v$parameter : informations sur les valeurs actuelles des paramètres
- v$session : informations sur les sessions courantes
- v$transaction: informations sur les transactions en cours
- - ...

- **Note**
  - Ce sont les tables dynamiques de gestion des performances
  - Ces tables commences par un v$.

EXERCICES Série 11

EXERCICES Série 11

11.1 "Quels sont les noms des colonnes de la table VOL ?" (voir : user_tab_columns, user_tables,...)

Notes :

-cols est un synonyme de user_tab_columns

11.2 "Quels sont les tables et les vues de votre schéma ?" (voir tab, cat, user_tables et user_views)

Notes :

-cat est un synonyme de user_catalog

-Tabletyp est le type de la colonne (une table, une vue...)

11.3 "Quelles sont les tables qui contiennent une colonne PL# ?"

11.4 "Quelles sont les vues du dictionnaire d'Oracle (voir DICT ou DICTIONARY) ? "

11.5 "Quels sont les tables appartenant à l'utilisateur SCOTT ?"

11.6 "Quels sont les contraintes existant dans votre schéma pour la table PILOTE ?"

5. Concurrence d'accès

5.1 Transaction

Définition

- Une transaction (unité logique de traitement) est une séquence d'instructions SQL qui doivent s'exécuter en totalité ou jamais.

5.1 Transaction

Début et fin d'une transaction Oracle

- A la connexion à un outil
- A la fin de la transaction précédente.
- Une transaction SQL *se termine* :
- Par un ordre COMMIT ou ROLLBACK
- Par un ordre du LDD valide :

  CREATE, DROP, RENAME, ALTER, ...

  ==> La transaction est validée : COMMIT ;
- A la déconnexion d'un outil : DISCONNECT, EXEC SQL, RELEASE).

  ==> La transaction est validée : COMMIT ;

  - lors d'une fin anormale du processus utilisateur.

  ==> La transaction est invalidée : ROLLBACK.

5.1 Transaction

Contrôle du déroulement d'une transaction

- Les clauses de contrôle du déroulement des transactions sont :

  COMMIT [WORK]

  SAVEPOINT savepoint_id

  ROLLBACK [WORK] [TO savepoint_id]

- **COMMIT :**
  - - valide l'ensemble des modifications depuis le début de la transaction
  - - libère les verrous

- **ROLLBACK :**
  - - restitue les données à leur valeur de début de transaction
  - - libère les verrous

- **ROLLBACK TO SAVEPOINT** :

  1. Pose d'étiquette : SAVEPOINT nom

  2 . Annulation partielle :

  ROLLBACK TO [SAVEPOINT] nom

  - **Note :**
    - l'utilisation de WORK est facultative
    - le paramètre SAVEPOINT dans *init.ora* fixe le nombre de points de sauvegardes : "savepoints" (par défaut 5).

EXERCICES Série 12

EXERCICES Série 12

```
-- T1 :
        INSERT INTO pilote
          values(25, 'Conficias', '19-SEP-42', 'Pekin', '13548254', 39000.0,null);
        select * from pilote where pl#=25;
        COMMIT ;
        select * from pilote where pl#=25;
-- T2 :
        UPDATE pilote SET plnom='Conficios' WHERE plnom='Conficias';
        select * from pilote where pl#=25;
        ROLLBACK ;
        select * from pilote where pl#=25;
--T3 :
        UPDATE pilote SET plnom='Conficies' WHERE plnom='Conficias';
        select * from pilote where pl#=25;

        SAVEPOINT updt_conf1;

        UPDATE pilote SET
        plnom='Conficius' WHERE plnom='Conficies';

        SAVEPOINT updt_conf2 ;
        select * from pilote where pl#=25;

        UPDATE pilote SET plnom='Conficios' WHERE plnom='Conficius';
        select * from pilote where pl#=25;

        ROLLBACK TO updt_conf1 ;
        select * from pilote where pl#=25;

        UPDATE pilote SET plnom='Conficius' WHERE plnom='Conficies';
        select * from pilote where pl#=25;

        UPDATE pilote SET sal=40000.0 WHERE plnom='Conficius';
        select * from pilote where pl#=25;

        COMMIT ;
        select * from pilote where pl#=25;
```

5.2 Gestion des verrous

Généralités

- Une des raisons d'être d'un SGBD **est l'accès concurrent** aux données par plusieurs utilisateurs.

- Aussi, pour assurer l'accès aux données sans risque d'anomalie de lecture ou de mise à jour, **Oracle utilise la technique du verrouillage**.

- Les verrous permettent ***d'éviter des CONFLITS entre des utilisateurs*** *qui tentent d'accéder à la même ressource.*

- La pose des verrous s'effectuent de deux façons :
  - **Implicitement**: c'est le moteur Oracle qui décide de poser un verrou ;
  - **Explicitement** : c'est le programmeur qui pose explicitement les verrous

- **Les granules de verrouillage sont** : *la table ou la ligne.*

5.2 Gestion des verrous

Verrous ligne (TX) :

- **Implicitement, un verrou exclusif est posé sur les lignes** concernées lors de l'utilisation de l'un des ordres SQL suivant :
  - INSERT ou
  - UPDATE ou
  - DELETE ou
  - SELECT ... FOR UPDATE

- Les autres utilisateurs ne pourront accéder à ces lignes jusqu'à la libération du dit verrou.

5.2 Gestion des verrous

Verrous table (TM) :

- Un VERROU TABLE permet :
  - d'éviter la pose de verrous CONTRAIRE aux actions de mises à jours déjà entreprises sur la table ou que l'on souhaite entreprendre sur la table
  - D'éviter qu'un autre utilisateur puisse **modifier ou supprimer la table** ... alors qu'elle est en cours de manipulation.
- Un VERROU TABLE peut être poser :
  - **Implicitement** lors de la pose d'un **VERROU LIGNE** lors de l'utilisation d'un des ordres SQL suivant :
    - INSERT ou
    - UPDATE ou
    - DELETE ou
    - SELECT ... FOR UPDATE
  - **Explicitement** avec l'utilisation de l'ordre SQL
    - LOCK TABLE

5.2 Gestion des verrous

Les différents types de Verrous sur les tables

- **X** : Verrou EXCLUSIVE. Ce verrou favorise la mise à jour de l'utilisateur l'ayant posé. Les autres utilisateurs ne peuvent poser aucun des autres verrous (S, RS, SRX, RX, X)

- **S** : Verrou SHARE favorise la lecture dans une table pour l'utilisateur l'ayant posé mais interdit les MISES A JOUR (X, SRX, RX) pour les autres

- **RS**: Verrou ROW SHARE (SHARE UPDATE) permet l'accès concurrent à une table. Aide à se prémunir d'un verrou X

- **RX**: Verrou ROW EXCLUSIVE se comporte comme RS mais interdit la pose de verrous S, SRX ou X

- **SRX**: Verrou SHARE ROW EXCLUSIVE. Favorise vos MAJ sans risquer S, SRX, RX ou X

5.2 Gestion des verrous

- **Le tableau ci-dessous indique les différents modes de verrouillages et les actions autorisées et interdites**

Commande SQL	Mode de verrouillage (niveau table)	Actions autorisées	Actions interdites
SELECT ... FROM table	aucun	Toutes On peut poser les verrous suivants : RS row share RX row exclusive S share SRX share row exclusive X exclusive	aucune
- UPDATE table, - INSERT INTO table, - DELETE FROM table, - LOCK TABLE table IN row exclusive mode	Row exclusif (RX)	- SELECT - INSERT - UPDATE - DELETE On peut poser les verrous suivants : RX, RS	Accès en mode exclusif en lecture et écriture: **S, SRX**, X avec les ordres : LOCK TABLE IN - share mode - share row exclusive - exclusive mode

5.2 Gestion des verrous

- **Le tableau ci-dessous indique les différents modes de verrouillages et les actions autorisées et interdites (suite)**

Commande SQL	Mode de verrouillage (niveau table)	Actions autorisées	Actions interdites
- SELECT ... FOR UPDATE, - LOCK TABLE table IN ROW SHARE MODE	Row Share (RS)	- SELECT - INSERT - UPDATE - DELETE On peut poser les verrous suivants : RX, RS, S, RSX	Accès en mode exclusif en écriture **X** avec l'ordre : LOCK TABLE In exclusive mode
Lock table *table* in share mode	Share (S)	- SELECT - SELECT ... FOR UPDATE On peut poser les verrous suivants : RS, S	- INSERT - UPDATE - DELETE Verrous : RX, SRX, **X** avec les ordres : LOCK TABLE IN - share row exclusive mode - exclusive mode - row exclusive mode

5.2 Gestion des verrous

- **Le tableau ci-dessous indique les différents modes de verrouillages et les actions autorisées et interdites (suite)**

Commande SQL	Mode de verrouillage (niveau table)	Actions autorisées	Actions interdites
lock table table in share row exclusive mode	Share Row Exclusif (SRX)	- SELECT - SELECT FOR UPDATE Verrous autorisés : RS	- INSERT - UPDATE - DELETE Verrous : RX, S, SRX, **X** avec les ordres : LOCK TABLE IN - share mode - share row exclusive mode - exclusive mode - row exclusive mode
lock table table in exclusive mode	Exclusif (X)	- SELECT Aucun verrou n'est autorisé	Tout sauf les requêtes.

5.2 Gestion des verrous

Verrous acquis selon les ordres SQL

Commande SQL	Verrous ligne	Mode de verrouillage de la table
SELECT	--	--
INSERT	oui	RX
UPDATE	oui	RX
DELETE	oui	RX
SELECT … FOR UPDATE …	oui	RS
LOCK TABLE nomTable IN ROW SHARE MODE	--	RS
LOCK TABLE nomTable IN SHARE ROW EXCLUSIVE MODE	--	SRX
LOCK TABLE nomTable IN SHARE MODE	--	S

5.2 Gestion des verrous

Verrous acquis selon les ordres SQL

Commande SQL	Verrous ligne	Mode de verrouillage de la table
LOCK TABLE nomTable IN EXCLUSIVE MODE	--	X
LOCK TABLE nomTable IN ROW EXCLUSIVE MODE		RX
DDL/DCL	--	X

5.2 Gestion des verrous

Pose explicite de verrous au niveau table

- La clause de verrouillage explicite au niveau d'une table est :

  LOCK TABLE <liste_de_table>

  IN <type_de_verrou>

  MODE [NOWAIT]

  (voir le tableau ci-dessus)

- **Note :**
  - Si un utilisateur A tente de verrouiller une table dans un mode incompatible avec un verrou posé par l'utilisateur B, il est mis en attente jusqu'à ce que B fasse COMMIT ou ROLLBACK (libère les verrous).
  - L'option NOWAIT permet d'éviter le blocage de A si la ressource n'est pas libre.

- Exemple :

  T1 (B) : LOCK TABLE pilote EXCLUSIF MODE ;

  T2 (A) : LOCK TABLE pilote shared row exclusif NOWAIT ;

5.2 Gestion des verrous

Pose implicite de verrous au niveau TABLE et LIGNES

- Les commandes SELECT FOR UPDATE, INSERT, DELETE, UPDATE placent un verrou exclusif sur une ou plusieurs lignes d'une table.

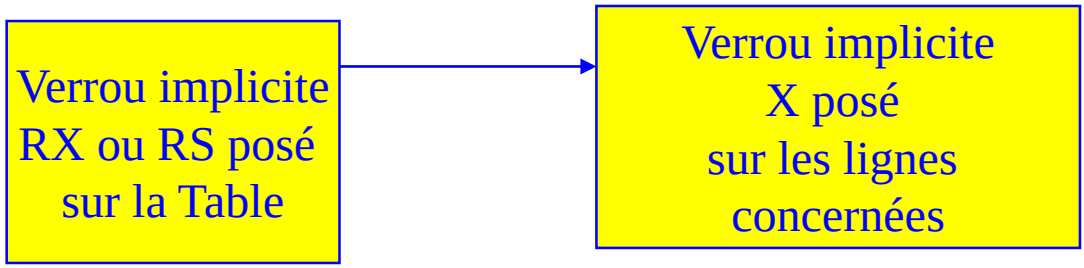
- Au niveau table, un verrou RS (row share) est posé pour la commande SELECT FOR UPDATE et un verrou RX (row exclusif). Un verrou ligne est toujours posé implicitement.

- Les verrous acquis grâce à la commande LOCK TABLE sont des verrous tables, ils ne verrouillent pas directement des lignes mais servent à se prémunir de certains verrous.

- Exemple
  - SELECT * FROM PILOTE
  - WHERE ADR='Paris'
  - FOR UPDATE OF SAL ;

5.2 Gestion des verrous

- **Pose implicite de verrous au niveau TABLE et LIGNES**



5.2 Gestion des verrous

Pose implicite de verrous au niveau TABLE et LIGNE

- Cette commande verrouille les lignes de tous les pilotes habitant Paris. Toutes les lignes sélectionnées sont verrouillées, pas seulement les champs apparaissant après OF.
  Ensuite on peut effectuer des mises à jours comme suit :

```
UPDATE PILOTE
SET SAL = 1.1 * SAL
WHERE ADR='Paris';
```

- *Note*

  – L'option FOR UPDATE ne s'emploie pas avec DISTINCT, GROUP BY, les opérateurs ensemblistes et les fonctions de groupes.

5.2 Gestion des verrous

Choix d'un type de verrou

- Le moteur Oracle pose des verrous implicites. Ceux-ci peuvent être remis en cause par l'utilisateur grâce aux commandes SELECT FOR UPDATE ou LOCK TABLE. Les verrous explicites pouvant influencer négativement les performances, leur pose doit se faire avec précautions :
  - Ne poser de verrous exclusifs sur une table que si le traitement l'exige ;
  - Poser les verrous de préférence au niveau ligne pour favoriser la concurrence ;
  - Poser un verrou share (S) pour favoriser les applications en lecture.

- Un verrou ligne favorise la concurrence mais pénalise les applications effectuant de nombreuses mises à jours. Dans le cas d'une application bancaire, la réalisation de l'opération DEBIT-CREDIT nécessite probablement un verrou EXCLUSIVE (X) au niveau table. La consultation de la table restant possible, les transactions de consultations de comptes se feront parallèlement.

EXERCICES Série 13

EXERCICES Série 13

- Pour effectuer ces tests il est nécessaire d'ouvrir deux sessions. Commentez les étapes où il y a un ?

Transaction 1	Temps	Transaction 2
LOCK TABLE pilote IN ROW SHARE MODE ;	1	
	2	DROP TABLE pilote ; ?
	3	LOCK TABLE pilote IN EXCLUSIVE MODE NOWAIT; ?
	4	select sal from pilote where pilote.adr= 'Paris' FOR UPDATE OF sal nowait ;
UPDATE pilote set sal = 12000.9 where adr = 'Paris'; (attente de T2)	5	
	6	ROLLBACK ; (libération des lignes verrouillées par T2)

EXERCICES Série 13

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
ROLLBACK; ?	7	
LOCK TABLE pilote IN ROW EXCLUSIVE MODE;	8	
	9	LOCK TABLE pilote IN EXCLUSIVE MODE NOWAIT; ?
	10	LOCK TABLE pilote IN SHARE ROW EXCLUSIVE MODE NOWAIT ; ?
	11	LOCK TABLE pilote IN SHARE MODE NOWAIT; ?
	12	UPDATE pilote set sal = 12000.9 where adr = 'Paris'; ?
	13	ROLLBACK; ?

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
SELECT sal FROM pilote WHERE adr='Paris' FOR UPDATE of sal; ?	14	
	15	UPDATE pilote set sal = 12000.9 where adr = 'Paris'; (attente T1)
ROLLBACK; ?	16	
	17	ROLLBACK; ?
LOCK TABLE pilote IN SHARE MODE ; ?	18	
	19	LOCK TABLE pilote IN EXCLUSIVE MODE NOWAIT ; ?
	20	LOCK TABLE pilote IN SHARE ROW EXCLUSIVE MODE NOWAIT ; ?

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
	21	LOCK TABLE pilote IN SHARE MODE nowait ; ?
	21a	LOCK TABLE pilote IN ROW EXCLUSIVE MODE nowait ; ?
	21b	LOCK TABLE pilote IN ROW SHARE MODE nowait; ?
	22	select sal from pilote where adr= 'Paris'; ?
	23	select sal from pilote where adr= 'Paris' FOR UPDATE OF sal nowait ; ?
	24	UPDATE pilote set sal = 12000.9 where adr = 'Paris'; (attente T1)
ROLLBACK ; ?	25	
	26	X lignes modifiées (après libération du verrou par T1) ROLLBACK ;
LOCK TABLE pilote IN SHARE ROW EXCLUSIVE MODE; ?	27	

EXERCICES Série 13

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
	28	LOCK TABLE pilote in EXCLUSIVE MODE NOWAIT; ?
	29	LOCK TABLE pilote in SHARE ROW EXCLUSIVE MODE NOWAIT; ?
	30	LOCK TABLE pilote in SHARE MODE NOWAIT; ?
	31	LOCK TABLE pilote in ROW EXCLUSIVE MODE NOWAIT; ?
	32	LOCK TABLE pilote IN ROW SHARE MODE; ?
	33	select sal from pilote where adr= 'Paris'; ?
	34	select sal from pilote where adr= 'Paris' FOR UPDATE OF sal nowait ; ?

EXERCICES Série 13

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
UPDATE pilote set sal = 12000.9 where adr = 'Paris';	35	UPDATE VOL SET VD='Nice' WHERE VOL#=310; ?
	36	UPDATE pilote set sal = 12000.9 where adr = 'Paris'; (attente T1)
UPDATE VOL SET VD='Toulouse' WHERE VOL#=310; (ATTENTE de T2)	37	-- Attention Verrou mortel
	38	(dead lock détecté, Transaction la plus récente est arrêtée) ROLLBACK;
ROLLBACK ?	39	
LOCK TABLE pilote IN EXCLUSIVE MODE; ?	40	
	41	LOCK TABLE pilote IN EXCLUSIVE MODE NOWAIT; ?

EXERCICES Série 13

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
	42	LOCK TABLE pilote IN SHARE ROW EXCLUSIVE MODE NOWAIT; ?
	43	LOCK TABLE pilote IN SHARE MODE NOWAIT; ?
	44	LOCK TABLE pilote IN ROW EXCLUSIVE MODE NOWAIT; ?
	45	LOCK TABLE pilote IN ROW SHARE MODE NOWAIT; ?
	46	select sal from pilote where adr= 'Paris'; ?
	47	select sal from pilote where adr= 'Paris' FOR UPDATE OF sal nowait;
UPDATE pilote set sal = 12000.9 where adr = 'Paris'; ?	48	

EXERCICES Série 13

EXERCICES Série 13

Transaction 1	Temps	Transaction 2
ROLLBACK;	49	
	50	ROLLBACK;

EXERCICES Série 13b

EXERCICES Série 13b

Ecrire une transaction SQL (à ne pas confondre avec les transactions bancaires) qui permet d'effectuer une opération de type DEBIT d'un compte et CREDIT d'un autre compte (on parlera aussi de virement) avec mise à jour des soldes.

Cas 1: verrous implicites uniquement

Cas 2 :Vous devez poser les verrous de façon appropriée avant de commencer les opérations sur les comptes. Le choix des verrous doit favoriser au maximum la concurrence d'accès en mise à jour

Cas 3: avec verrous explicites mais on doit débiter un compte épargne et créditer un compte chèque. Attention: le compte épargne ne doit pas devenir négatif

6.1. Les vues

- 6. Les vues
  - 6.1. Création de vues
  - 6.2. Manipulation sur les vues

6.1. Création d'une vue

Généralités

- 1. Une vue est une table logique qui permet l'accès aux données de une ou plusieurs tables de bases de façon transparente.

- 2. Une vue ne contient aucune ligne. Les données sont stockées dans les tables.

- 3. Les vues sont utilisées pour :
  - Assurer *l'indépendance logique/externe* ;
  - Fournir un *niveau supplémentaire de sécurité* sur les tables de base. Ainsi on peut restreindre, pour un utilisateur donné, l'accès à quelques. lignes d'une table
  - M*asquer la complexité* : une vue peut être la jointure de N-tables ;
  - Fournir *une nouvelle vision de la base*. Au lieu de changer le nom des tables de base, on changera seulement au niveau d'une vue si le changement ne concerne pas toute les applications ;
  - M*asquer les bases de données distantes*.

6.1. Création d'une vue

Syntaxe

CREATE [OR REPLACE]
[FORCE | NOFORCE] VIEW nom_de_vue
[(alias_colonne1, alias_colonne2, ...)] AS subquery
[WITH {CHECK OPTION [CONSTRAINT constraint] |
READ ONLY}];

- **Mots clés et paramètres**

OR REPLACE : permet de supprimer puis de recréer la vue si elle existe

FORCE : ignore les erreurs et crée la vue

WITH CHECK OPTION : permet d'assurer la cohérence des informations modifiées afin de laisser dans la vue les lignes affectées par une modification

CONSTRAINT constraint : permet juste de nommer la contrainte de la vue

READ ONLY : Pas de mises à jour à travers la vue

6.1. Création d'une vue

Remarques

- La modification d'une table de base affecte la vue
- Le corps d'une vue ne peut contenir de clause FOR UPDATE
- on ne peut effectuer des insertions, des mises à jours et des suppressions dans une vue contenant une jointure (certains cas uniquement), des opérateurs ensemblistes, des fonctions de groupe, les clauses GROUP BY, CONNECT BY ou START WITH et l'opérateur DISTINCT.
- Les vues systèmes: **all_views, dba_views, user_views** contiennent des informations sur les vues
- Les vues systèmes : **all_updatable_columns, user_updatable_columns, dba_updatable_columns** donnent des informations sur les colonnes modifiables dans une vue

6.1. Création d'une vue

Exemple

- 1. Création d'une vue permettant d'accéder au pilote qui habite Nice

```
CREATE OR REPLACE VIEW V_PILOTE_NICE
AS SELECT * FROM PILOTE WHERE ADR='Nice';
```

Il sera possible de consulter, insérer des lignes, supprimer des lignes et modifier des lignes dans la table PILOTE à travers cette vue.

On pourra créer, supprimer ou modifier des pilote n'habitant pas Nice.

- 2. Recréer la même vue qu'en 1 pour n'autoriser que les mises à jours sur les pilotes habitant Nice

```
CREATE OR REPLACE VIEW V_PILOTE_NICE
AS SELECT * FROM PILOTE WHERE ADR='Nice'
WITH CHECK OPTION CONSTRAINT
    chk_pilote_adr;
```

6.1. Création d'une vue

Exemple

- 3. Recréer la même vue qu'en 1 en interdisant toute mise à jour

```
CREATE OR REPLACE VIEW V_PILOTE_NICE3
AS SELECT * FROM PILOTE WHERE ADR='Nice'
WITH READ ONLY;
```

- 4. Création d'une vue contenant une colonne virtuelle salaire annuel (sal_Annuel)

```
CREATE OR REPLACE VIEW V_PILOTE
(PL# , PLNOM, DNAISS, ADR, TEL, SAL, sal_annuel)
AS SELECT pilote.*, sal*12 FROM PILOTE ;
Insert v_pilote  (PL# , PLNOM, DNAISS, ADR, TEL,
     SAL )values(30, 'Toto'  )
```

6.1. Création d'une vue

Exemple

- 5. Création d'une vue complexe donnant des informations sur les pilote, les vols et les avions en service

```
CREATE OR REPLACE VIEW V_PILOTE_EN_SERV
(VOL#, VD, HD, DAT, PLNOM, AVTYPE)
AS SELECT VOL#, VD, HD, DAT, PLNOM, AVTYPE
FROM PILOTE, AVION, VOL
WHERE PL#=PILOTE# AND AV#=AVION# ;

DELETE FROM V_PILOTE_EN_SERV
WHERE plnom='Mopolo';
```

6.2. Manipulation sur les vues

Généralités

- 1. Une vue est manipulable, comme une table de base, avec les clauses SQL (SELECT, INSERT, DELETE, UPDATE).

- 2. Une vue peut servir à la construction de requêtes imbriquées puisqu'elle peut apparaître derrière la clause FROM.

- **Opérations autorisées**
  - **Vue avec jointure** :
    Delete : OUI* Update : OUI * Insert : OUI *
  - **Vue avec Group By ou Distinct** :
    Delete : NON Update : NON Insert : NON
  - **Vue avec référence à RowNum** :
    Delete : NON Update : NON Insert : NON
  - **Vue avec une colonne issue d'une expression** :
    Delete : OUI Update : OUI sur autres col. Insert : OUI sur autre col.
  - **Vue avec au moins une colonne NOT NULL absente** :
    Delete : OUI Update : OUI Insert NON**
  - * : Uniquement pour les colonnes modifiables (voir la vue {dba | all | user}_updatable_columns). Ce sont les colonnes des tables pour lesquelles un identifiant unique existe dans la vue
  - ** sauf si une valeur par défaut existe sur la colonne

EXERCICES Série 14

Indépendance logique/externe : *vue de sélection*

14.1 - *"Créer une vue AVA300 qui donne tous les A300 de la compagnie"*

14.2 - *"Que se passe - t-il à l'insertion d'un "B707" dans la vue ?"*

Indépendance logique/externe : *renommage et ré-ordonnancement des colonnes*

14.3 - *"Créer une vue PAYE qui donne pour chaque pilote son salaire mensuel et annuel"*

14.4 - *"Créer une vue AVPLUS qui donne tous les numéros d'avions conduits par plus d'un pilote."*

14.5 - *"Créer une vue PILPARIS qui donne les noms, les numéros de vols, des pilotes qui assurent un vol au départ de Paris"*

Création d'une vue pour assurer la confidentialité

14.6 *"Créer une vue PILSANS qui donne les renseignements concernant les pilotes, sans le salaire."*

EXERCICES Série 14

EXERCICES Série 14

Vues issues d'une table

14.7 *"Créer une vue AVIONNICE : Ensemble des avions localisés à Nice"*

Modification à travers une vue

1) Lister l'extension de la vue AVIONNICE

2) Mise à jour d'un tuple dans cette vue : localiser l'avion de n° 5 à Paris

3) Mise à jour d'un tuple dans cette vue : localiser l'avion n° 7 à Nice

4) Lister la table de base AVION. Que constatez-vous ?

Insertion dans la vue

1) Insérer le tuple (11, 'A300', 220, 'Nice', 'EN service');

2) lister l'extension de la vue AVIONNICE

3) lister la table de base AVION.

Suppression dans la vue

1) Suppression de l'avion N° 11

2) lister l'extension de la vue AVIONNICE

3) lister la table AVION.

EXERCICES Série 14

EXERCICES Série 14

Vues issues de plusieurs tables

14.8 "Créer une vue AVSERVPARIS : Ensemble des avions en service localisés à Paris"

Modification de la vue

1) lister l'extension de la vue AVSERVPARIS

2) mise à jour d'un tuple de cette vue. Que remarque-t-on ?"

Insertion dans la vue

1) recréez la vue avec jointure

2) insertion d'un tuple dans la vue AVSERVPARIS. Que remarque-t-on?

suppression dans la vue

1) suppression de tous les pilotes de n° inférieur à 7 dans AVSERVPARIS

Vues contenant des colonnes virtuelles

14.9 "Reprendre la vue PAYE et lister son contenu"

Modification via la vue

1) Mise à jour d'un tuple dans cette vue : mettre le salaire du pilote 1 à 0

2) lister le contenu de cette vue. Que remarque--on ?

Insertion via la vue

1) insertion d'un tuple dans la vue PAYE . Que remarque-t-on ?

Suppression via la vue

1) suppression de tous les pilotes dont le salaire annuel est supérieur à 180000.

EXERCICES Série 14

EXERCICES Série 14

Vues contenant une clause GROUP BY

"Reprenons la vue AVPLUS. Llister cette vue"

"Quels sont le n° d'avions conduits par plus d'un pilote et localisés à Paris ?

7. Annexes

Annexe 1

VERROUS IMPLICITES ET EXPLICITES

Actions Transaction T1 de l'utilisateur U1 :

INSERT INTO pilote VALUES(10, 'Math', '12-AUG-1938', 'Paris', '23548254', 15000.0);
Update pilote set sal=12000 where sal between 21000 and 22000;

Actions Transaction T2 de l'utilisateur U2 :

Delete From pilote where PL#=4 ;

Actions Transaction T3 de l'utilisateur U3 :

Lock table pilote in exclusive mode ; -- Verrou explicite

Niveau table : T1(RX) T1(RX) T2 (RX attente) T3(X attente)

PL#	PLNOM	DNAISS	ADR	TEL	SAL
---	---	---	---	---	---

Niveau lignes

	1	Miranda	16/08/52	Sophia-Antipolis	93548254	18009
	2	St-exupery	16/10/32	Lyon	91548254	12300
	3	Armstrong	11/03/30	Wapakoneta	96548254	24500
T2(x attente)						
T1(x)	4	Tintin	01/08/29	Bruxelles	93548254	~~21100~~ 12000
	5	Gagarine	12/08/34	Klouchino	93548454	22100
T1(x)	6	Baudry	31/08/59	Toulouse	93548444	~~21000~~ 12000
T1(x)	8	Bush	28/02/24	Milton	44556254	~~22000~~ 12000
T1(x)	9	Ruskoi	16/08/30	Moscou	73548254	~~22000~~ 12000
T1(x)	10	Math	12/08/38	Paris	23548254	15000

7. Annexes

Annexe 1

Validation de Transaction T1 de l'utilisateur U1 :

INSERT INTO pilote VALUES(10, 'Math', '12-AUG-1938', 'Paris', '23548254', 15000.0);
Update pilote set sal=12000 where sal between 21000 and 22000;
Commit;

Actions Transaction T2 de l'utilisateur U2 :

Delete From pilote where PL#=4 ;

Actions Transaction T3 de l'utilisateur U3 :

Lock table pilote in exclusive mode ; -- Verrou explicite

Niveau table : T2 (RX) T3(X attente)

PL#	PLNOM	DNAISS	ADR	TEL	SAL
---	---	---	---	---	---

Niveau lignes

	1	Miranda	16/08/52	Sophia-Antipolis	93548254	18009
	2	St-exupery	16/10/32	Lyon	91548254	12300
	3	Armstrong	11/03/30	Wapakoneta	96548254	24500
T2(x)	4	Tintin	01/08/29	Bruxelles	93548254	12000
	5	Gagarine	12/08/34	Klouchino	93548454	22100
	6	Baudry	31/08/59	Toulouse	93548444	12000
	8	Bush	28/02/24	Milton	44556254	12000
	9	Ruskoi	16/08/30	Moscou	73548254	12000
	10	Math	12/08/38	Paris	23548254	15000