

# COURS

## Base de donnée

Pr H.Moutachaouik  
ENSAM CASABLANCA

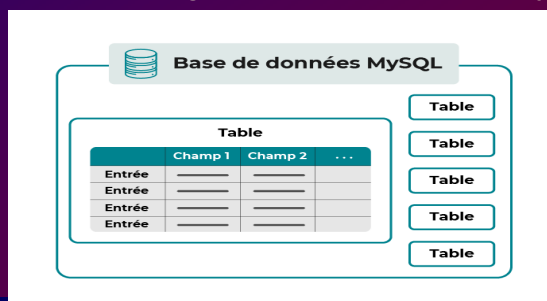
[hicham.moutachaouik@ensam-casa.ma](mailto:hicham.moutachaouik@ensam-casa.ma)

1-1

## Base de donnée

**Une base de donnée:**

- est une collection d'informations organisées afin d'être facilement consultables, gérables et mises à jour.



1-2

## Plan

```
SELECT      [DISTINCT] {*, column [alias], ...}
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

01) L'ordre  
SELECT  
élémentaire

- Select, from
- expression arithmétique
- Problème de null dans le calcul
- Alias
- Concaténation || ou concat
- Doublons (distinct)

2) Sélection et  
Tri des Lignes

- Where
- Chaînes de Caractères et Dates
- Opérateurs de Comparaison
- Recherche des valeurs NULL
- Opérateurs Logiques (AND, OR, NOT)
- Règles de Priorité (AND, OR, NOT, .....)
- ORDER BY

1-3

## Plan

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column1 = table2.column2;
```

03) Fonctions  
Mono-Ligne

- Caractère
  - Lower, upper, initcap
  - Concat, substr, length, instr, lpad
- Numérique : round, mod
- Date
- Conversion: convert
- If, case
- Imbrication des Fonctions

04) Afficher des  
Données  
Issues de  
Plusieurs Tables  
(jointure)

- Equijointure
- Jointure externe
- Non-équijointure
- Autojointure

1-4

## Plan

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

05) Fonctions de Groupe && Groupement

- AVG, COUNT, MAX, MIN, SUM
- Divisez une table en groupes de lignes avec **GROUP BY**

06) Opérateurs Ensemblistes

- Union / Union All
- Intersect
- Minus

1-5

## Plan

```
SELECT      select_list
FROM table
WHERE       expr operator
            (SELECT      select_list
             FROM        table);
```

07) Sous-Interrogations

- Sous-interrogation mono-ligne
- Sous-interrogation multi-ligne
- Sous-interrogation multi-colonne
- Utilisation d'une Sous-Interrogation dans la Clause FROM

08) Manipulation des Données

- Décrire chaque ordre du LMD
- Insérer des lignes dans une table
- Mettre à jour des lignes dans une table
- Supprimer des lignes d'une table

1-6

## Plan

```
CREATE TABLE [schema.] table  
    (column datatype [DEFAULT expr], ...
```

09) Création et  
Gestion de  
Tables

- CREATE TABLE
- Création d'une Table au Moyen d'une Sous-Interrogation
- ALTER TABLE (ADD, MODIFY, CHANGE)
- DROP TABLE

10) Contraintes  
et Vues

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CREATE VIEW

Synthèse

- Synthèse de la formation

1-7

## Langage SQL

1-8

# 1

## L'Ordre SELECT Elémentaire

### Objectifs

**A la fin de ce chapitre, vous saurez :**

- **Enumérer toutes les possibilités de l'ordre SQL SELECT**
- **Exécuter un ordre SELECT élémentaire**

## Les Possibilités de l'Ordre SQL SELECT

### Sélection


Table 1

### Projection


Table 1

### Jointure


Table 1


Table 2

1-11

## Ordre SELECT Élémentaire

```
SELECT    [DISTINCT] {*, column [alias],...}
FROM      table;
```

- **SELECT** indique **quelles** colonnes rapporter
- **FROM** indique dans **quelle** table rechercher

1-12

## Ecriture des Ordres SQL

- Les ordres SQL peuvent être écrits indifféremment en majuscules et/ou minuscules.
- Les ordres SQL peuvent être écrits sur plusieurs lignes.
- Les mots-clés ne doivent pas être abrégés ni scindés sur deux lignes différentes.
- Les tabulations et indentations permettent une meilleure lisibilité.

1-13

## Sélection de Toutes les Colonnes

```
SQL> SELECT *  
2 FROM dept;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

1-14

## Sélection d'Une ou Plusieurs Colonnes Spécifiques

```
SQL> SELECT deptno, loc  
2 FROM dept;
```

DEPTNO	LOC
10	NEW YORK
20	DALLAS
30	CHICAGO
40	BOSTON

1-15

## Valeurs par Défaut des En-têtes de Colonne

- Justification par défaut
  - A gauche : date et données alphanumériques
  - A droite : données numériques
- Affichage par défaut : en majuscules

1-16



## Expressions Arithmétiques

Possibilité de créer des expressions avec des données de type **NUMBER** et **DATE** au moyen d'opérateurs arithmétiques

Opérateur	Description
+	Addition
-	Soustraction
*	Multiplication
/	Division

1-17

## Utilisation des Opérateurs Arithmétiques

```
SQL> SELECT ename, sal, sal+300  
2 FROM emp;
```

ENAME	SAL	SAL+300
-----	-----	-----
KING	5000	5300
BLAKE	2850	3150
CLARK	2450	2750
JONES	2975	3275
MARTIN	1250	1550
ALLEN	1600	1900
...		
14 rows selected.		

1-18

## Priorité des Opérateurs

**\* / + -**

- La multiplication et la division ont priorité sur l'addition et la soustraction.
- A niveau de priorité identique, les opérateurs sont évalués de gauche à droite.
- Les parenthèses forcent la priorité d'évaluation et permettent de clarifier les ordres.

1-19

## Priorité des Opérateurs

```
SQL> SELECT ename, sal, 12*sal+100  
2 FROM emp;
```

ENAME	SAL	12*SAL+100
KING	5000	60100
BLAKE	2850	34300
CLARK	2450	29500
JONES	2975	35800
MARTIN	1250	15100
ALLEN	1600	19300
...		

14 rows selected.

1-20

## Utilisation des Parenthèses

```
SQL> SELECT ename, sal, 12*(sal+100)
2 FROM emp;
```

ENAME	SAL	12*(SAL+100)
KING	5000	61200
BLAKE	2850	35400
CLARK	2450	30600
JONES	2975	36900
MARTIN	1250	16200
...		

14 rows selected.

1-21

## La Valeur NULL

- NULL représente une valeur non disponible, non affectée ou inconnue,
- La valeur NULL est différente du zéro ou de l'espace.

```
SQL> SELECT ename, job, comm
2 FROM emp;
```

ENAME	JOB	COMM
KING	PRESIDENT	
BLAKE	MANAGER	
...		
TURNER	SALESMAN	0
...		

14 rows selected.

1-22

## Valeurs NULL dans les Expressions Arithmétiques

Les expressions arithmétiques comportant une valeur NULL sont évaluées à NULL

```
SQL> select ename , 12*sal+comm  
2   from emp  
3  WHERE ename='KING' ;
```

ENAME	12*SAL+COMM
-----	-----
KING	

1-23

## L'Alias de Colonne

- Renomme un en-tête de colonne
- Est utile dans les calculs
- Suit immédiatement le nom de la colonne ; le mot-clé AS placé entre le nom et l'alias est optionnel
- Doit obligatoirement être inclus entre guillemets s'il contient des espaces ou des caractères spéciaux

1-24

## Utilisation des Alias de Colonnes

```
SQL> SELECT ename AS name, sal salary  
2 FROM emp;
```

NAME	SALARY
-----	-----
...	

```
SQL> SELECT ename "Name",  
2 sal*12 "Annual Salary"  
3 FROM emp;
```

Name	Annual Salary
-----	-----
...	

1-25

## L'Opérateur de Concaténation

- Concatène des colonnes ou chaînes de caractères avec d'autres colonnes
- Est représenté par deux barres verticales (||) ou concat
- La colonne résultante est une expression caractère

1-26

## Utilisation de l'Opérateur de Concaténation

```
SQL> SELECT  ename || job AS "Employees"
      2 FROM    emp;
```

```
Employees
-----
KINGPRESIDENT
BLAKEMANAGER
CLARKMANAGER
JONESMANAGER
MARTINSALESMAN
ALLENSALESMAN
...
14 rows selected.
```

1-27

## Utilisation de l'Opérateur de Concaténation

```
SQL> SELECT  concat(ename,job) AS "Employees"
      2 FROM    emp;
```

```
Employees
-----
KINGPRESIDENT
BLAKEMANAGER
CLARKMANAGER
JONESMANAGER
MARTINSALESMAN
ALLENSALESMAN
...
14 rows selected.
```

1-28

## Littéral

- Un littéral est un caractère, une expression, ou un nombre inclus dans la liste SELECT.
- Chaque littéral apparaît sur chaque ligne ramenée.

1-29

## Utilisation des Chaînes de Caractères Littérales

```
SQL> SELECT ename || ' is a ' || job  
2          AS "Employee Details"  
3 FROM    emp;
```

```
Employee Details  
-----  
KING is a PRESIDENT  
BLAKE is a MANAGER  
CLARK is a MANAGER  
JONES is a MANAGER  
MARTIN is a SALESMAN  
...  
14 rows selected.
```

1-30

## Doublons

Par défaut, le résultat d'une requête affiche toutes les lignes, y compris les doublons.

```
SQL> SELECT deptno  
2 FROM emp;
```

DEPTNO
10
30
10
20
...

14 rows selected.

1-31

## Elimination des Doublons

Pour éliminer les doublons il faut ajouter le mot-clé DISTINCT à la clause SELECT.

```
SQL> SELECT DISTINCT deptno  
2 FROM emp;
```

DEPTNO
10
20
30

1-32



## Contrôle des acquis

### Base de données exemple

**EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)**

**DEPT (DEPTNO, DNAME, LOC)**

1-33

## Contrôle des acquis

**Exprimer en SQL les requêtes suivantes.**

- a. Sélectionnez toutes les données de la table DEPT .**
- b. Créez une requête pour afficher le nom , le poste , la date d'embauche et le matricule de chaque employé, en plaçant le matricule en premier.**
- c. Créez une requête pour afficher les différents types de poste existant dans la table EMP.**
- d. Nom , salaire, commission, salaire+commission de tous les employés.**

### Base de données exemple

**EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)**

**DEPT (DEPTNO, DNAME, LOC)**

1-34

# 2

## Sélection et Tri des Lignes Retournées par un SELECT

### Objectifs

**A la fin de ce chapitre, vous saurez :**


- **Limiter le nombre de lignes retournées par une requête**
- **Trier les lignes retournées par une requête**

## Sélectionner les Lignes

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

“...rechercher  
tous les employés  
du département  
10”



EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7782	CLARK	MANAGER		10
7934	MILLER	CLERK		10

## Sélectionner les Lignes

- Restreindre la sélection au moyen de la clause **WHERE**.

```
SELECT      [DISTINCT] {*, column [alias], ...}  
FROM        table  
[WHERE      condition(s)];
```

- La clause **WHERE** se place après la clause **FROM**.

## Utilisation de la Clause WHERE

```
SQL> SELECT ename, job, deptno  
2 FROM emp  
3 WHERE job='CLERK';
```

ENAME	JOB	DEPTNO
-----	-----	-----
JAMES	CLERK	30
SMITH	CLERK	20
ADAMS	CLERK	20
MILLER	CLERK	10

## Chaînes de Caractères et Dates

- Les constantes chaînes de caractères et dates doivent être placées entre simples quotes.
- La recherche tient compte des majuscules et minuscules (pour les chaînes de caractère) et du format (pour les dates.)

```
SQL> SELECT ename, job, deptno  
2 FROM emp  
3 WHERE ename = 'JAMES';
```

## Opérateurs de Comparaison

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

## Utilisation des Opérateurs de Comparaison

```
SQL> SELECT ename, sal, comm
2 FROM emp
3 WHERE sal<=comm;
```

ENAME	SAL	COMM
-----	-----	-----
MARTIN	1250	1400

## Autres Opérateurs de Comparaison

Opérateur	Signification
BETWEEN ...AND...	Compris entre ... et ... (bornes comprises)
IN (liste)	Correspond à une valeur de la liste
LIKE	Ressemblance partielle de chaînes de caractères
IS NULL	Correspond à une valeur NULL

## Utilisation de l'Opérateur BETWEEN

**BETWEEN** permet de tester l'appartenance à une fourchette de valeurs.

```
SQL> SELECT  ename, sal
2 FROM      emp
3 WHERE     sal BETWEEN 1000 AND 1500;
```

ENAME	SAL	Limite inférieure	Limite supérieure
MARTIN	1250		
TURNER	1500		
WARD	1250		
ADAMS	1100		
MILLER	1300		

## Utilisation de l'Opérateur IN

IN permet de comparer une expression avec une liste de valeurs.

```
SQL> SELECT empno, ename, sal, mgr
2 FROM emp
3 WHERE mgr IN (7902, 7566, 7788);
```

EMPNO	ENAME	SAL	MGR
7902	FORD	3000	7566
7369	SMITH	800	7902
7788	SCOTT	3000	7566
7876	ADAMS	1100	7788

## Utilisation de l'Opérateur LIKE

- LIKE permet de rechercher des chaînes de caractères à l'aide de caractères génériques
- Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux.
  - (%) représente zéro ou plusieurs caractères
  - ( \_ ) représente un caractère

```
SQL> SELECT ename
2 FROM emp
3 WHERE ename LIKE 'S%';
```

## Utilisation de l'Opérateur LIKE

- Vous pouvez combiner plusieurs caractères génériques de recherche.

```
SQL> SELECT  ename
2  FROM      emp
3  WHERE     ename LIKE '_A%';
```

ENAME
-----
JAMES
WARD

## Utilisation de l'Opérateur IS NULL

Recherche de valeurs NULL avec  
l'opérateur IS NULL

```
SQL> SELECT  ename, mgr
2  FROM      emp
3  WHERE     mgr IS NULL;
```

ENAME	MGR
-----	-----
KING	



## Opérateurs Logiques

Opérateur	Signification
AND	Retourne TRUE si <i>les deux</i> conditions sont VRAIES
OR	Retourne TRUE si <i>l'une au moins</i> des conditions est VRAIE
NOT	Ramène la valeur TRUE si la condition qui suit l'opérateur est FAUSSE

## Utilisation de l'Opérateur AND

Avec AND, les deux conditions doivent être VRAIES.

```
SQL> SELECT empno, ename, job, sal
2  FROM emp
3  WHERE sal>=1100
4  AND job='CLERK';
```

EMPNO	ENAME	JOB	SAL
7876	ADAMS	CLERK	1100
7934	MILLER	CLERK	1300

## Utilisation de l'Opérateur OR

Avec OR, l'une ou l'autre des deux conditions doit être VRAIE.

```
SQL> SELECT empno, ename, job, sal
2 FROM emp
3 WHERE sal >= 1100
4 OR job = 'CLERK';
```

EMPNO	ENAME	JOB	SAL
7839	KING	PRESIDENT	5000
7698	BLAKE	MANAGER	2850
7782	CLARK	MANAGER	2450
7566	JONES	MANAGER	2975
7654	MARTIN	SALESMAN	1250
...			

14 rows selected.

## Utilisation de l'Opérateur NOT

```
SQL> SELECT ename, job
2 FROM emp
3 WHERE job NOT IN ('CLERK', 'MANAGER', 'ANALYST');
```

ENAME	JOB
KING	PRESIDENT
MARTIN	SALESMAN
ALLEN	SALESMAN
TURNER	SALESMAN
WARD	SALESMAN

```
... WHERE sal NOT BETWEEN 1000 AND 1500
... WHERE ename NOT LIKE '%A%'
... WHERE comm IS NOT NULL
```

## Règles de Priorité

Ordre de priorité	Opérateur
1	Tous les opérateurs de comparaison
2	NOT
3	AND
4	OR

Les parenthèses permettent de modifier les règles de priorité

## Règles de Priorité

```
SQL> SELECT ename, job, sal
2 FROM emp
3 WHERE job='SALESMAN'
4 OR job='PRESIDENT'
5 AND sal>1500;
```

ENAME	JOB	SAL
-----	-----	-----
KING	PRESIDENT	5000
MARTIN	SALESMAN	1250
ALLEN	SALESMAN	1600
TURNER	SALESMAN	1500
WARD	SALESMAN	1250

## Règles de Priorité

Utilisation de parenthèses pour forcer la priorité.

```
SQL> SELECT  ename, job, sal
2 FROM      emp
3 WHERE      (job='SALESMAN'
4 OR         job='PRESIDENT')
5 AND        sal>1500;
```

ENAME	JOB	SAL
KING	PRESIDENT	5000
ALLEN	SALESMAN	1600

## Clause ORDER BY

- Tri des lignes avec la clause ORDER BY
  - ASC : ordre croissant (par défaut)
  - DESC : ordre décroissant
- La clause ORDER BY se place à la fin de l'ordre SELECT

```
SQL> SELECT  ename, job, deptno, hiredate
2 FROM      emp
3 ORDER BY  hiredate;
```

ENAME	JOB	DEPTNO	HIREDATE
SMITH	CLERK	20	17-DEC-80
ALLEN	SALESMAN	30	20-FEB-81
...			

14 rows selected.

## Tri par Ordre Décroissant

```
SQL> SELECT   ename, job, deptno, hiredate
2  FROM      emp
3  ORDER BY   hiredate DESC;
```

ENAME	JOB	DEPTNO	HIREDATE
ADAMS	CLERK	20	12-JAN-83
SCOTT	ANALYST	20	09-DEC-82
MILLER	CLERK	10	23-JAN-82
JAMES	CLERK	30	03-DEC-81
FORD	ANALYST	20	03-DEC-81
KING	PRESIDENT	10	17-NOV-81
MARTIN	SALESMAN	30	28-SEP-81
...			

14 rows selected.

## Tri sur l'Alias de Colonne

```
SQL> SELECT   empno, ename, sal*12 annsal
2  FROM      emp
3  ORDER BY   annsal;
```

EMPNO	ENAME	ANNSAL
7369	SMITH	9600
7900	JAMES	11400
7876	ADAMS	13200
7654	MARTIN	15000
7521	WARD	15000
7934	MILLER	15600
7844	TURNER	18000
...		

14 rows selected.

## Tri sur Plusieurs Colonnes

- L'ordre des éléments de la liste ORDER BY donne l'ordre du tri.

```
SQL> SELECT  ename, deptno, sal  
2 FROM      emp  
3 ORDER BY  deptno, sal DESC;
```

ENAME	DEPTNO	SAL
KING	10	5000
CLARK	10	2450
MILLER	10	1300
FORD	20	3000
...		

14 rows selected.

- Vous pouvez effectuer un tri sur une colonne ne figurant pas dans la liste SELECT.

## Résumé

```
SELECT      [DISTINCT] {*, column [alias], ...}  
FROM        table  
[WHERE      condition(s)]  
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

## Résumé

```
SELECT      [DISTINCT] {*, column [alias], ...}  
FROM        table  
[WHERE      condition(s)]  
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

3-1

## 3

## Fonctions Mono-Ligne

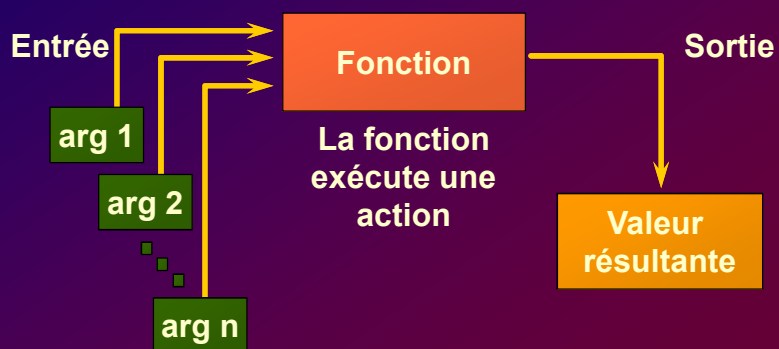
## Objectifs

A la fin de ce chapitre, vous saurez :

- Décrire différents types de fonctions SQL
- Utiliser les fonctions caractère, numériques et date dans les ordres SELECT
- Expliquer les fonctions de conversion

3-3

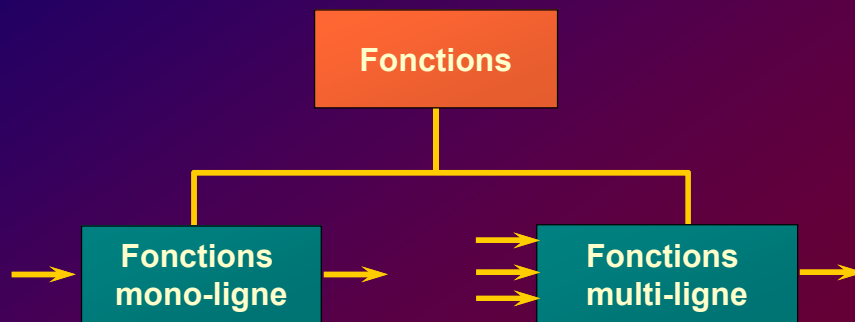
## Fonctions SQL



3-4



## Deux Types de Fonctions SQL



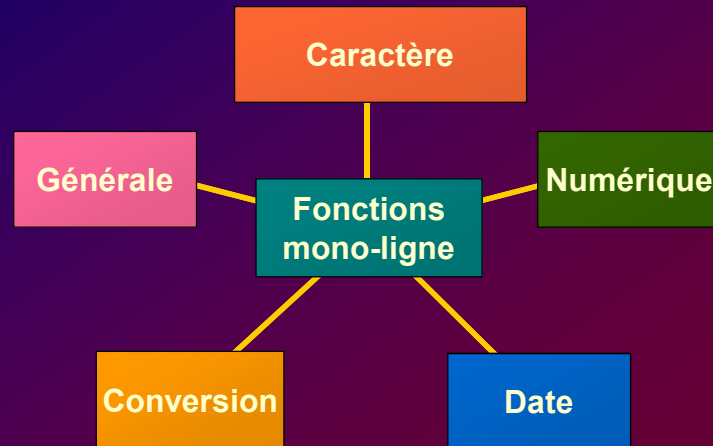
3-5

## Fonctions Mono-Ligne

- Manipulent des éléments de données
- Acceptent des arguments et ramènent une valeur
- Agissent sur chacune des lignes rapportées
- Ramènent un seul résultat par ligne
- Peuvent modifier les types de données
- Peuvent être imbriquées

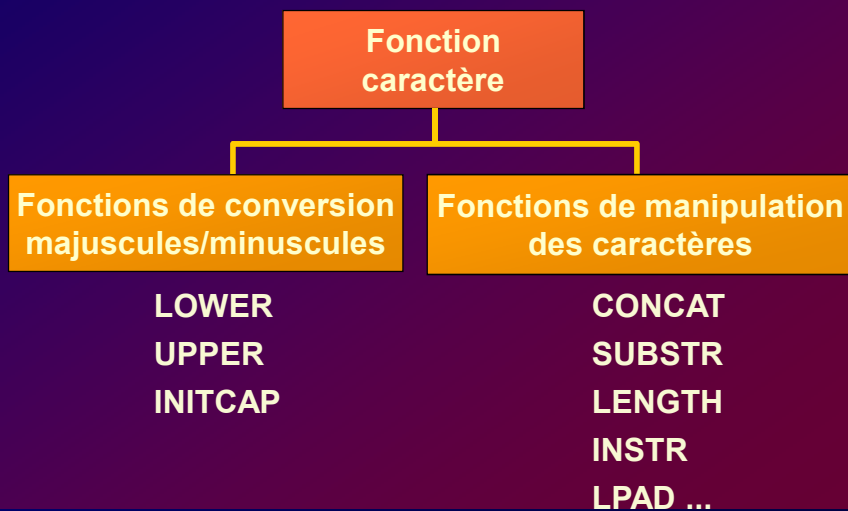
3-6

## Fonctions Mono-Ligne



3-7

## Fonctions Caractère



3-8

## Fonctions de Conversion Majuscules/Minuscules

Fonction	Résultat
LOWER('Cours SQL')	cours sql
UPPER('Cours SQL')	COURS SQL
INITCAP('Cours SQL')	Cours Sql

3-9

## Utilisation des Fonctions de Conversion Majuscules/Minuscules

Afficher le matricule, le nom et le numéro de département de l'employé Blake.

```
SQL> SELECT empno, ename, deptno
2 FROM emp
3 WHERE ename = 'blake';
no rows selected
```

```
SQL> SELECT empno, ename, deptno
2 FROM emp
3 WHERE LOWER(ename) = 'blake';
```

EMPNO	ENAME	DEPTNO
7698	BLAKE	30

3-10

## Fonctions de Manipulation des Caractères

### Manipulation de chaînes de caractères

Fonction	Résultat
CONCAT('Une', 'Chaîne')	UneChaîne
SUBSTR('Chaîne',1,3)	Cha
LENGTH('Chaîne')	6
INSTR('Chaîne', 'a')	3
LPAD(sal,10,'*')	*****5000

3-11

## Utilisation des Fonctions de Manipulation des Caractères

```
SQL> SELECT ename, CONCAT (ename, job), LENGTH (ename),
2          INSTR (ename, 'A')
3 FROM emp
4 WHERE SUBSTR (job,1,5) = 'SALES';
```

ENAME	CONCAT (ENAME, JOB)	LENGTH (ENAME)	INSTR (ENAME, 'A')
MARTIN	MARTINSALESMAN	6	2
ALLEN	ALLENSALESMAN	5	1
TURNER	TURNERSALESMAN	6	0
WARD	WARDSALESMAN	4	2

3-12

## Fonctions Numériques

- **ROUND** : Cette fonction permet soit d'arrondir sans utiliser de décimal pour retourner un nombre entier (c'est-à-dire : aucun chiffre après la virgule), ou de choisir le nombre de chiffre après la virgule.

**ROUND(45.926)** → 46

**ROUND(45.926, 2)** → 45.93

**ROUND(45.923, 2)** → 45.92

- **MOD** : Ramène le reste d'une division

**MOD(1600, 300)** → 100

3-13

## Utilisation de la Fonction MOD

Calculer le reste de la division salaire par commission pour l'ensemble des employés ayant un poste de vendeur.

```
SQL> SELECT  ename, sal, comm, MOD(sal, comm)
2 FROM      emp
3 WHERE     job = 'SALESMAN';
```

ENAME	SAL	COMM	MOD (SAL, COMM)
MARTIN	1250	1400	1250
ALLEN	1600	300	100
TURNER	1500	0	1500
WARD	1250	500	250

3-14

## Opérations Arithmétiques sur les Dates

- Ajout ou soustraction d'un nombre à une date pour obtenir un résultat de type **date**.
- Soustraction de deux dates afin de déterminer le **nombre** de jours entre ces deux dates.

3-15

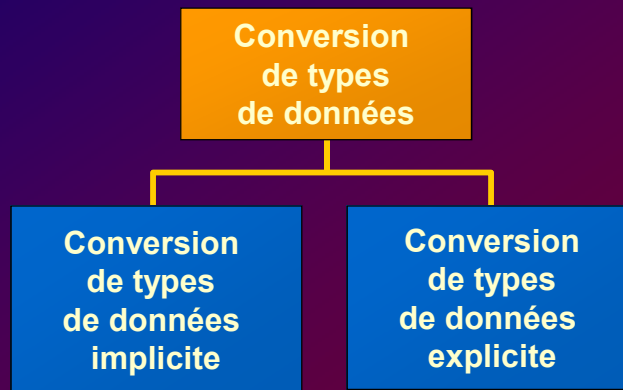
## Utilisation d'Opérateurs Arithmétiques avec les Dates

```
SQL> SELECT ename, DATEDIFF(SYSDATE(),hiredate)/7 WEEKS  
2 FROM emp  
3 WHERE deptno = 10;
```

ENAME	WEEKS
-----	-----
KING	830.93709
CLARK	853.93709
MILLER	821.36566

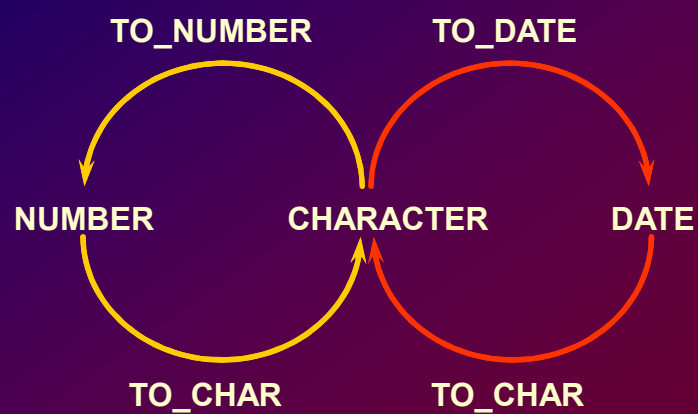
3-16

## Fonctions de Conversion



3-17

## Conversion de Types de Données Explicite



3-18

## Modèles de Format Date

YYYY	Année exprimée avec 4 chiffres
YEAR	Année exprimée en toutes lettres
MM	Mois exprimé avec 2 chiffres
MONTH	Mois exprimé en toutes lettres
DY	3 premières lettres du nom du jour
DAY	Jour exprimé en toutes lettres

3-19

## Utilisation de la Fonction TO\_CHAR avec les Dates

```
SQL> SELECT ename,
2         TO_CHAR(hiredate, 'fmDD Month YYYY') HIREDATE
3 FROM emp;
```

ENAME	HIREDATE
KING	17 November 1981
BLAKE	1 May 1981
CLARK	9 June 1981
JONES	2 April 1981
MARTIN	28 September 1981
ALLEN	20 February 1981
...	

14 rows selected.

3-20



## Fonction NVL

Convertit une valeur NULL en une valeur réelle

- Fonctionne avec les données de type date, caractère et numérique.
- Les types de données doivent correspondre
  - NVL(comm,0)
  - NVL(hiredate,'01-JAN-97')
  - NVL(job,'No Job Yet')

3-21

## Utilisation de la Fonction NVL

```
SQL> SELECT ename, sal, comm, (sal*12)+NVL(comm,0)
2 FROM emp;
```

ENAME	SAL	COMM	(SAL*12)+NVL(COMM,0)
KING	5000		60000
BLAKE	2850		34200
CLARK	2450		29400
JONES	2975		35700
MARTIN	1250	1400	16400
ALLEN	1600	300	19500
...			
14 rows selected.			

3-22

## Fonction case

### Facilite les recherches conditionnelles

```
CASE
  WHEN condition1 THEN result1
  WHEN condition2 THEN result2
  WHEN conditionN THEN resultN
  ELSE result
END;
```

```
SELECT OrderID, Quantity,
```

```
CASE
```

```
  WHEN Quantity > 30 THEN "The quantity is greater than 30"
```

```
  WHEN Quantity = 30 THEN "The quantity is 30"
```

```
  ELSE "The quantity is under 30"
```

```
END
```

3-23

```
FROM OrderDetails;
```

## Fonction DECODE

### Facilite les recherches conditionnelles en jouant le rôle de **CASE** ou **IF-THEN-ELSE**

```
DECODE(col/expression, search1, result1
      [, search2, result2,...,]
      [, default])
```

3-24

## Utilisation de la Fonction DECODE dans oracle

```
SQL> SELECT job, sal,
2      DECODE(job, 'ANALYST', SAL*1.1,
3              'CLERK',   SAL*1.15,
4              'MANAGER', SAL*1.20,
5                      SAL)
6      REVISÉD_SALARY
7 FROM emp;
```

JOB	SAL	REVISÉD_SALARY
PRESIDENT	5000	5000
MANAGER	2850	3420
MANAGER	2450	2940
...		

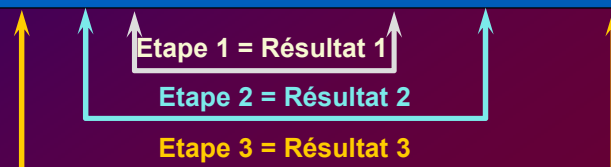
14 rows selected.

3-25

## Imbrication des Fonctions

- Le niveau d'imbrication des fonctions mono-ligne est illimité
- Les fonctions imbriquées sont évaluées de l'intérieur vers l'extérieur

**F3 (F2 (F1 (col, arg1) , arg2) , arg3)**



3-26

## Imbrication des Fonctions

```
SQL> SELECT  ename,  
2          NVL (TO_CHAR(mgr) , 'No Manager')  
3 FROM emp  
4 WHERE     mgr IS NULL;
```

ENAME	NVL (TO_CHAR (MGR) , 'NOMANAGER' )
KING	No Manager

## Afficher des Données Issues de Plusieurs Tables

ORACLE®

## Afficher des Données Issues de Plusieurs Tables

EMP

EMPNO	ENAME	...	DEPTNO
7839	KING	...	10
7698	BLAKE	...	30
...			
7934	MILLER	...	10

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

↓ ↓

EMPNO	DEPTNO	LOC
7839	10	NEW YORK
7698	30	CHICAGO
7782	10	NEW YORK
7566	20	DALLAS
7654	30	CHICAGO
7499	30	CHICAGO
...		

14 rows selected.

ORACLE®

## Qu'est-ce qu'une Jointure ?

Une jointure sert à extraire des données de plusieurs tables.

```
SELECT  table1.column, table2.column  
FROM    table1, table2  
WHERE   table1.column1 = table2.column2;
```

- Ecrivez la condition de jointure dans la clause WHERE.
- Placez le nom de la table avant le nom de la colonne lorsque celui-ci figure dans plusieurs tables.

ORACLE®

## Produit Cartésien

- On obtient un produit cartésien lorsque :
  - Une condition de jointure est omise
  - Une condition de jointure est incorrecte
- Toutes les lignes de la première table sont jointes à toutes les lignes de la seconde
- Pour éviter un produit cartésien, toujours insérer une condition de jointure correcte dans la clause WHERE.

ORACLE®

## Génération d'un Produit Cartésien

EMP (14 lignes)

EMPNO	ENAME	...	DEPTNO
7839	KING	...	10
7698	BLAKE	...	30
...			
7934	MILLER	...	10

DEPT (4 lignes)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

"Produit  
cartésien :  
14\*4=56 lignes"

ENAME	DNAME
KING	ACCOUNTING
BLAKE	ACCOUNTING
...	
KING	RESEARCH
BLAKE	RESEARCH
...	
56 rows selected.	

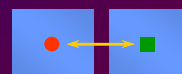
ORACLE®

## Types de Jointures

Equijointure



Non-équijointure



Jointure externe



Autojointure



ORACLE®

## Qu'est-ce qu'une Equijointure ?

EMP

EMPNO	ENAME	DEPTNO
7839	KING	10
7698	BLAKE	30
7782	CLARK	10
7566	JONES	20
7654	MARTIN	30
7499	ALLEN	30
7844	TURNER	30
7900	JAMES	30
7521	WARD	30
7902	FORD	20
7369	SMITH	20
...		

14 rows selected.

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
30	SALES	CHICAGO
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
30	SALES	CHICAGO
20	RESEARCH	DALLAS
20	RESEARCH	DALLAS
...		

14 rows selected.

Clé étrangère Clé primaire

ORACLE®

## Extraction d'Enregistrements avec les Equijointures

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
2          dept.deptno, dept.loc
3 FROM emp, dept
4 WHERE emp.deptno=dept.deptno;
```

EMPNO	ENAME	DEPTNO	DEPTNO	LOC
7839	KING	10	10	NEW YORK
7698	BLAKE	30	30	CHICAGO
7782	CLARK	10	10	NEW YORK
7566	JONES	20	20	DALLAS
...				

14 rows selected.

ORACLE®



## Différencier les Noms de Colonne Ambigus

- Préfixer avec le nom de la table pour différencier les noms de colonnes appartenant à plusieurs tables.
- Ces préfixes de table améliorent les performances.
- Différencier des colonnes de même nom appartenant à plusieurs tables en utilisant des alias de colonne.

ORACLE®

## Ajout de Conditions de Recherche avec l'Opérateur AND

EMP			DEPT		
EMPNO	ENAME	DEPTNO	DEPTNO	DNAME	LOC
7839	KING	10	10	ACCOUNTING	NEW YORK
7698	BLAKE	30	30	SALES	CHICAGO
7782	CLARK	10	10	ACCOUNTING	NEW YORK
7566	JONES	20	20	RESEARCH	DALLAS
7654	MARTIN	30	30	SALES	CHICAGO
7499	ALLEN	30	30	SALES	CHICAGO
7844	TURNER	30	30	SALES	CHICAGO
7900	JAMES	30	30	SALES	CHICAGO
7521	WARD	30	30	SALES	CHICAGO
7902	FORD	20	20	RESEARCH	DALLAS
7369	SMITH	20	20	RESEARCH	DALLAS
...			...		
14 rows selected.			14 rows selected.		

ORACLE®

## Utilisation d'Alias de Table

Simplifiez les requêtes avec les alias de table.

```
SQL> SELECT emp.empno, emp.ename, emp.deptno,
2      dept.deptno, dept.loc
3 FROM   emp, dept
4 WHERE  emp.deptno=dept.deptno;
```

```
SQL> SELECT e.empno, e.ename, e.deptno,
2      d.deptno, d.loc
3 FROM   emp e, dept d
4 WHERE  e.deptno=d.deptno;
```

ORACLE®

## Jointures de Plus de Deux Tables

CUSTOMER		ORD			
NAME	CUSTID	CUSTID	ORDID		
-----	-----	-----	-----		
JOCKSPORTS	100	101	610		
TKB SPORT SHOP	101	102	611		
VOLLYRITE	102	104	612		
JUST TENNIS	103	106	601		
K+T SPORTS	105	102	602		
SHAPE UP	106	106			
WOMENS SPORTS	107	106			
...	...	...			
9 rows selected.		21 rows			

ORDID	ITEMID
-----	-----
610	3
611	1
612	1
601	1
602	1
...	
64 rows selected.	

ORACLE®

## Non-Equijointures

EMP

EMPNO	ENAME	SAL
7839	KING	5000
7698	BLAKE	2850
7782	CLARK	2450
7566	JONES	2975
7654	MARTIN	1250
7499	ALLEN	1600
7844	TURNER	1500
7900	JAMES	950
...		

14 rows selected.

SALGRADE

GRADE	LOSAL	HISAL
1	700	1200
2	1201	1400
3	1401	2000
4	2001	3000
5	3001	9999

"Les salaires (SAL) de la table EMP sont compris entre le salaire minimum (LOSAL) et le salaire maximum (HISAL) de la table SALGRADE"

ORACLE®

## Extraction d'Enregistrements avec les Non-Equijointures

```
SQL> SELECT e.ename, e.sal, s.grade
2 FROM emp e, salgrade s
3 WHERE e.sal
4 BETWEEN s.losal AND s.hisal;
```


ENAME	SAL	GRADE
...		
JAMES	950	1
SMITH	800	1
ADAMS	1100	1
...		

14 rows selected.

ORACLE®

## Jointures Externes

EMP		DEPT	
ENAME	DEPTNO	DEPTNO	DNAME
-----	-----	-----	-----
KING	10	10	ACCOUNTING
BLAKE	30	30	SALES
CLARK	10	10	ACCOUNTING
JONES	20	20	RESEARCH
...		...	
		40	OPERATIONS


**Pas d'employés dans le département OPERATIONS**

ORACLE®

## Jointures Externes

- Les jointures externes permettent de visualiser des lignes qui ne répondent pas à la condition de jointure.
- L'opérateur de jointure externe est le signe (+).

```
SELECT table.column, table.column
FROM   table1, table2
WHERE  table1.column (+) = table2.column;
```

```
SELECT table.column, table.column
FROM   table1, table2
WHERE  table1.column = table2.column (+);
```

ORACLE®

## Utilisation des Jointures Externes

```
SQL> SELECT  e.ename, d.deptno, d.dname
2  FROM      emp e, dept d
3  WHERE      e.deptno(+) = d.deptno
4  ORDER BY  e.deptno;
```

ENAME	DEPTNO	DNAME
-----	-----	-----
KING	10	ACCOUNTING
CLARK	10	ACCOUNTING
...		
	40	OPERATIONS

15 rows selected.

ORACLE®

## Autojointures

EMP

EMPNO	ENAME	MGR
-----	-----	-----
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698



"Dans la table WORKER, MGR équivaut à EMPNO  
dans la table MANAGER"

ORACLE®

## Autojointures

EMP (WORKER)

EMPNO	ENAME	MGR
7839	KING	
7698	BLAKE	7839
7782	CLARK	7839
7566	JONES	7839
7654	MARTIN	7698
7499	ALLEN	7698

EMP (MANAGER)

EMPNO	ENAME
7839	KING
7839	KING
7839	KING
7698	BLAKE
7698	BLAKE



"Dans la table WORKER, MGR équivaut à EMPNO dans la table MANAGER"

ORACLE®

## Liaison d'une Table à Elle-même

```
SQL> SELECT worker.ename || ' works for ' || manager.ename
2 FROM emp worker, emp manager
3 WHERE worker.mgr = manager.empno;
```

```
WORKER.ENAME || 'WORKSFOR' || MANAG
-----
BLAKE works for KING
CLARK works for KING
JONES works for KING
MARTIN works for BLAKE
...
13 rows selected.
```

ORACLE®

# 5

## Regrouper les Données avec les Fonctions de Groupe

### Objectifs

**A la fin de ce chapitre, vous saurez :**

- Identifier et expliquer les fonctions de groupe disponibles
- Regrouper les données avec la clause **GROUP BY**
- Inclure ou exclure des groupes de lignes avec la clause **HAVING**

## Fonctions de Groupe

Les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe.

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

"salaire maximum  
de la table EMP"

MAX (SAL)
5000

5-3

## Types de Fonctions de Groupe

- AVG()
- COUNT ()
- MAX ()
- MIN ()
- SUM ()

5-4



## Fonctions AVG et SUM

AVG et SUM s'utilisent avec des données numériques.

```
SQL> SELECT AVG(sal), MAX(sal),  
2          MIN(sal), SUM(sal)  
3 FROM emp  
4 WHERE job LIKE 'SALES%';
```

AVG (SAL)	MAX (SAL)	MIN (SAL)	SUM (SAL)	
1400	1600	1250	5600	

5-5

## Fonctions MIN et MAX

MIN et MAX s'utilisent avec tous types de données.

```
SQL> SELECT MIN(hiredate), MAX(hiredate)  
2 FROM emp;
```

MIN (HIRED)	MAX (HIRED)	
17-DEC-80	12-JAN-83	

5-6

## Utilisation de la Fonction COUNT

**COUNT(\*)** ramène le nombre de lignes d'une table.

```
SQL> SELECT COUNT (*)  
2 FROM emp  
3 WHERE deptno = 30;
```

COUNT (*)
----- 6

5-7

## Utilisation de la Fonction COUNT

**COUNT(expr)** ramène le nombre de lignes non NULL.

```
SQL> SELECT COUNT (comm)  
2 FROM emp  
3 WHERE deptno = 30;
```

COUNT (COMM)
----- 4

5-8

## Fonctions de Groupe et Valeurs NULL

Les fonctions de groupe ignorent les valeurs NULL des colonnes.

```
SQL> SELECT AVG(comm)
       2 FROM emp;
```

AVG (COMM)
-----
550

5-9

## Utilisation de la Fonction NVL avec les Fonctions de Groupe

La fonction NVL force la prise en compte des valeurs NULL dans les fonctions de groupe.

```
SQL> SELECT AVG(NVL(comm,0))
       2 FROM emp;
```

AVG (NVL (COMM, 0))
-----
157.14286

5-10

## Création de Groupes de Données

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

2916.6667

2175

1566.6667

"salaire  
moyen pour  
chaque  
département  
de la table  
EMP"

DEPTNO	AVG (SAL)
10	2916.6667
20	2175
30	1566.6667

5-11

## Création de Groupes de Données : la Clause GROUP BY

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY  group_by_expression]
[ORDER BY  column];
```

Divisez une table en groupes de lignes  
avec la clause GROUP BY.

5-12

## Utilisation de la Clause GROUP BY

La clause GROUP BY doit inclure toutes les colonnes de la liste SELECT qui ne figurent pas dans des fonctions de groupe.

```
SQL> SELECT deptno, AVG(sal)
2 FROM emp
3 GROUP BY deptno;
```

DEPTNO	AVG(SAL)
10	2916.6667
20	2175
30	1566.6667

5-13

## Utilisation de la Clause GROUP BY

La colonne citée en GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

```
SQL> SELECT AVG(sal)
2 FROM emp
3 GROUP BY deptno;
```

AVG(SAL)
2916.6667
2175
1566.6667

5-14

## Regroupement sur Plusieurs Colonnes

EMP

DEPTNO	JOB	SAL
10	MANAGER	2450
10	PRESIDENT	5000
10	CLERK	1300
20	CLERK	800
20	CLERK	1100
20	ANALYST	3000
20	ANALYST	3000
20	MANAGER	2975
30	SALESMAN	1600
30	MANAGER	2850
30	SALESMAN	1250
30	CLERK	950
30	SALESMAN	1500
30	SALESMAN	1250

"somme des salaires de la table EMP pour chaque poste, regroupés par département"

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
20	MANAGER	2975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600

5-15

## Utilisation de la Clause GROUP BY sur Plusieurs Colonnes

```
SQL> SELECT deptno, job, sum(sal)
2 FROM emp
3 GROUP BY deptno, job;
```

DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20	ANALYST	6000
20	CLERK	1900
...		

9 rows selected.

5-16

## Erreurs d'utilisation des Fonctions de Groupe dans une Requête

- Vous ne pouvez utiliser la clause WHERE pour limiter les groupes.
- Utilisez la clause HAVING.

```
SQL> SELECT deptno, AVG(sal)
2 FROM emp
3 WHERE AVG(sal) > 2000
4 GROUP BY deptno;
```

```
WHERE AVG(sal) > 2000
*
ERROR at line 3:
ORA-00934: group function is not allowed here
```

5-17

## Exclusion de Groupes

EMP

DEPTNO	SAL
10	2450
10	5000
10	1300
20	800
20	1100
20	3000
20	3000
20	2975
30	1600
30	2850
30	1250
30	950
30	1500
30	1250

5000

3000

2850

"salaire maximum  
supérieur à  
\$2900 dans  
chaque  
département"

DEPTNO	MAX (SAL)
10	5000
20	3000

5-18

## Exclusion de Groupes : la Clause HAVING

Utilisez la clause HAVING pour restreindre les groupes

- Les lignes sont regroupées.
- La fonction de groupe est appliquée.
- Les groupes qui correspondent à la clause HAVING sont affichés.

```
SELECT      column, group_function
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[HAVING     group_condition]
[ORDER BY   column];
```

5-19

## Utilisation de la clause HAVING

```
SQL> SELECT deptno, max(sal)
2 FROM emp
3 GROUP BY deptno
4 HAVING max(sal)>2900;
```

DEPTNO	MAX (SAL)
10	5000
20	3000

5-20



## Utilisation de la Clause HAVING

```
SQL> SELECT      job, SUM(sal) PAYROLL
  2  FROM        emp
  3  WHERE       job NOT LIKE 'SALES%'
  4  GROUP BY    job
  5  HAVING      SUM(sal)>5000
  6  ORDER BY    SUM(sal);
```

JOB	PAYROLL
ANALYST	6000
MANAGER	8275

5-21

## Imbrication des Fonctions de Groupe

Afficher le salaire moyen maximum.

```
SQL> SELECT      max(avg(sal))
  2  FROM        emp
  3  GROUP BY    deptno;
```

MAX (AVG (SAL))
2916.6667

5-22

## Résumé

```
SELECT      column, group_function  
FROM        table  
[WHERE      condition]  
[GROUP BY   group_by_expression]  
[HAVING     group_condition]  
[ORDER BY   column];
```

# 7

## Sous-Interrogations

### Objectifs

**A la fin de ce chapitre, vous saurez :**

- **Décrire les types de problèmes que les sous-interrogations peuvent résoudre**
- **Définir des sous-interrogations**
- **Enumérer les types de sous-interrogations**

## Utilisation d'une Sous-Interrogation pour Résoudre un Problème

"Qui a un salaire supérieur à celui de Jones ?"

Requête principale



"Quel employé a un salaire supérieur à celui de Jones ?"

sous-interrogation



"Quel est le salaire de Jones ?"

## Sous-Interrogations

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT  select_list
         FROM    table);
```

- La sous-interrogation (requête interne) est exécutée une fois avant la requête principale.
- Le résultat de la sous-interrogation est utilisé par la requête principale (externe).

## Utilisation d'une Sous-Interrogation

```
SQL> SELECT  ename
2  FROM      emp
3  WHERE     sal > 2975
4             (SELECT sal
5              FROM    emp
6              WHERE   empno=7566) ;
```

ENAME

-----

KING

FORD

SCOTT

## Conventions d'Utilisation des Sous-Interrogations

- Placez les sous-interrogations entre parenthèses.
- Placez les sous-interrogations à droite de l'opérateur de comparaison.
- N'ajoutez jamais de clause ORDER BY à une sous-interrogation.
- Utilisez les opérateurs mono-ligne avec les sous-interrogations mono-ligne.
- Utilisez les opérateurs multi-ligne avec les sous-interrogations multi-ligne.

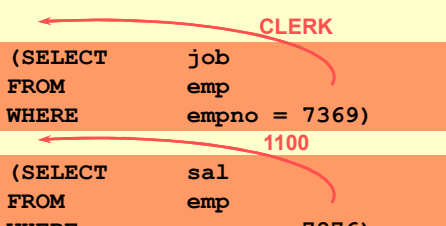
## Sous-Interrogations Mono-ligne

- Ne ramènent qu'une seule ligne
- Utilisent des opérateurs de comparaison mono-ligne

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

## Exécution de Sous-Interrogations Mono-ligne


```
SQL> SELECT  ename, job
2 FROM      emp
3 WHERE     job =
4           (SELECT  job
5            FROM      emp
6            WHERE     empno = 7369)
7 AND       sal >
8           (SELECT  sal
9            FROM      emp
10           WHERE     empno = 7876) ;
```



ENAME	JOB
-----	-----
MILLER	CLERK

## Utilisation de Fonctions de Groupe dans une Sous-Interrogation

```
SQL> SELECT  ename, job, sal
2 FROM      emp
3 WHERE     sal =
4           (SELECT MIN(sal)
5            FROM     emp);
```

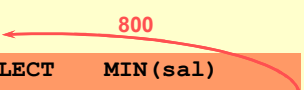


ENAME	JOB	SAL
-----	-----	-----
SMITH	CLERK	800

## Clause HAVING avec Sous-Interrogations

- Oracle Server exécute les sous-interrogations en premier.
- Oracle Server ramène les résultats dans la clause HAVING de la requête principale.

```
SQL> SELECT      deptno, MIN(sal)
2 FROM          emp
3 GROUP BY      deptno
4 HAVING        MIN(sal) >
5              (SELECT MIN(sal)
6               FROM     emp
7               WHERE    deptno = 20);
```



## Qu'est-ce Qui ne Va pas dans cet Ordre ?

```
SQL> SELECT empno, ename
2  FROM emp
3  WHERE sal =
4
5          (SELECT MIN(sal)
6  FROM emp
  GROUP BY deptno);
```

```
ERROR:
ORA-01427: single-row sub-query returns more than
one row
no rows selected
```

## Sous-Interrogation Multi-ligne

- Ramène plusieurs lignes
- Utilise des opérateurs de comparaison multi-ligne

Opérateur	Signification
IN	Egal à un élément quelconque de la liste
ANY	Compare la valeur à chaque valeur ramenée par la sous-interrogation
ALL	Compare la valeur à toutes les valeurs ramenées par la sous-interrogation



## Utilisation de l'Opérateur ANY dans les Sous-Interrogations Multi-ligne

```
SQL> SELECT empno, ename, job
2 FROM emp
3 WHERE sal < ANY
4 (SELECT sal
5 FROM emp
6 WHERE job = 'CLERK')
7 AND job <> 'CLERK';
```

EMPNO	ENAME	JOB
-----	-----	-----
7654	MARTIN	SALESMAN
7521	WARD	SALESMAN

## Utilisation de l'Opérateur ALL dans les Sous-Interrogations Multi-ligne

```
SQL> SELECT empno, ename, job
2 FROM emp
3 WHERE sal > ALL
4 (SELECT avg(sal)
5 FROM emp
6 GROUP BY deptno)
```

EMPNO	ENAME	JOB
-----	-----	-----
7839	KING	PRESIDENT
7566	JONES	MANAGER
7902	FORD	ANALYST
7788	SCOTT	ANALYST

# **Manipulation des Données**

## **Objectifs**

**A la fin de ce chapitre, vous saurez :**

- **Décrire chaque ordre du LMD**
- **Insérer des lignes dans une table**
- **Mettre à jour des lignes dans une table**
- **Supprimer des lignes d'une table**
- **Contrôler les transactions**

## Langage de Manipulation des Données

- Un ordre du LMD est exécuté lorsque :
  - Vous ajoutez des lignes à une table
  - Vous modifiez des lignes existantes dans une table
  - Vous supprimez des lignes d'une table

## Ajout d'une Nouvelle Ligne dans une Table

50	DEVELOPMENT	DETROIT
----	-------------	---------

Nouvelle ligne

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

"...insérer une nouvelle ligne dans la table DEPT ..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT

## L'Ordre INSERT

- L'ordre INSERT permet d'ajouter de nouvelles lignes dans une table.

```
INSERT INTO      table [(column [, column...])]  
VALUES          (value [, value...]);
```

- Cette syntaxe n'insère qu'une seule ligne à la fois.

## Insertion de Nouvelles Lignes

- Insérez une nouvelle ligne en précisant une valeur pour chaque colonne.
- Eventuellement, énumérez les colonnes dans la clause INSERT.

```
SQL> INSERT INTO    dept (deptno, dname, loc)  
2  VALUES          (50, 'DEVELOPMENT', 'DETROIT');  
1 row created.
```

- Indiquez les valeurs dans l'ordre par défaut des colonnes dans la table.
- Placez les valeurs de type caractère et date entre simples quotes.

## Insertion de Lignes Contenant des Valeurs NULL

- Méthode implicite : ne spécifiez pas la colonne dans la liste.

```
SQL> INSERT INTO dept (deptno, dname)
2 VALUES (60, 'MIS');
1 row created.
```

- Méthode explicite : spécifiez le mot-clé NULL.

```
SQL> INSERT INTO dept
2 VALUES (70, 'FINANCE', NULL);
1 row created.
```

## Insertion de Valeurs Spéciales

La fonction SYSDATE renvoie la date et l'heure courantes.

```
SQL> INSERT INTO emp (empno, ename, job,
2 mgr, hiredate, sal, comm,
3 deptno)
4 VALUES (7196, 'GREEN', 'SALESMAN',
5 7782, SYSDATE, 2000, NULL,
6 10);
1 row created.
```

## Copie de Lignes d'une Autre Table

- Ecrivez votre ordre INSERT en spécifiant une sous-interrogation.

```
SQL> INSERT INTO managers(id, name, salary, hiredate)
2      SELECT empno, ename, sal, hiredate
3      FROM    emp
4      WHERE   job = 'MANAGER';
3 rows created.
```

- N'utilisez pas la clause VALUES.
- Le nombre de colonnes de la clause INSERT doit correspondre à celui de la sous-interrogation.

## Modification des Données d'une Table

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		10
7566	JONES	MANAGER		20
...				

"...modifier une ligne de la table EMP..."

EMP

EMPNO	ENAME	JOB	...	DEPTNO
7839	KING	PRESIDENT		10
7698	BLAKE	MANAGER		30
7782	CLARK	MANAGER		20
7566	JONES	MANAGER		20
...				

## L'Ordre UPDATE

- Utilisez l'ordre UPDATE pour modifier des lignes existantes.

```
UPDATE      table
SET         column = value [, column = value]
[WHERE      condition];
```

- Si nécessaire, vous pouvez modifier plusieurs lignes à la fois.

## Modification de Lignes d'une Table

- La clause WHERE permet de modifier une ou plusieurs lignes spécifiques.

```
SQL> UPDATE emp
2 SET deptno = 20
3 WHERE empno = 7782;
1 row updated.
```

- Si vous omettez la clause WHERE, toutes les lignes sont modifiées.

```
SQL> UPDATE employee
2 SET deptno = 20;
14 rows updated.
```

## Modification de Lignes en Fonction d'une Autre Table

Utilisez des sous-interrogations dans l'ordre UPDATE pour modifier des lignes d'une table à l'aide de valeurs d'une autre table.

```
SQL> UPDATE   employee
2  SET      deptno = (SELECT   deptno
3                        FROM     emp
4                        WHERE    empno = 7788)
5  WHERE    job      = (SELECT   job
6                        FROM     emp
7                        WHERE    empno = 7788) ;
2 rows updated.
```

## Suppression d'une Ligne d'une Table

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	DEVELOPMENT	DETROIT
60	MIS	
...		

"...supprime une ligne de la table DEPT..."

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
60	MIS	
...		



## L'Ordre DELETE

**Vous pouvez supprimer des lignes d'une table au moyen de l'ordre DELETE.**

```
DELETE [FROM]   table
[WHERE          condition];
```

## Suppression de Lignes d'une Table

- La clause WHERE permet de supprimer une ou plusieurs lignes spécifiques.

```
SQL> DELETE FROM    department
      2 WHERE         dname = 'DEVELOPMENT';
1 row deleted.
```

- Si vous omettez la clause WHERE, toutes les lignes sont supprimées.

```
SQL> DELETE FROM    department;
4 rows deleted.
```

## Suppression de Lignes en Faisant Référence à une Autre Table

Utilisez des sous-interrogations dans l'ordre DELETE pour supprimer des lignes dont certaines valeurs correspondent à celles d'une autre table.

```
SQL> DELETE FROM      employee
2   WHERE      deptno =
3               (SELECT  deptno
4                  FROM    dept
5                  WHERE   dname = 'SALES') ;
6 rows deleted.
```

## Suppression de Lignes : Erreur de Contrainte d'Intégrité

```
SQL> DELETE FROM      dept
2   WHERE      deptno = 10;
```

```
DELETE FROM dept
*
ERROR at line 1:
ORA-02292: integrity constraint (USR.EMP_DEPTNO_FK)
violated - child record found
```

*Vous ne pouvez pas supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table.*

## Résumé

Ordre	Description
INSERT	Ajoute une nouvelle ligne dans une table
UPDATE	Modifie des lignes dans une table
DELETE	Supprime des lignes d'une table

# **Création et Gestion de Tables**

## **Objectifs**

**A la fin de ce chapitre, vous saurez :**

- **Décrire les principaux objets d'une base de données**
- **Créer des tables**
- **Décrire les différents types de données utilisables pour les définitions de colonne**
- **Modifier la définition des tables**
- **Supprimer, renommer et tronquer une table**

## Objets d'une Base de Données

Objet	Description
Table	Unité de stockage élémentaire, composée de lignes et de colonnes
Vue	Représente de manière logique des sous-groupes de données issues d'une ou plusieurs tables

## L'Ordre CREATE TABLE

- Vous devez posséder :
  - Un privilège CREATE TABLE
  - Un espace de stockage

```
CREATE TABLE [schema.] table  
    (column datatype [DEFAULT expr], ...
```

- Spécifiez :
  - Un nom de table
  - Le nom, le type de données et la taille des colonnes.

## Création de Tables

- Créer la table.

```
SQL> CREATE TABLE dept
2      (deptno int(2) ,
3      dname  VARCHAR(14) ,
4      loc    VARCHAR(13) ) ;
Table created.
```

- Vérifier la création de la table.

```
SQL> DESCRIBE dept
```

Name	NULL?	Type
-----	-----	-----
DEPTNO		int(2)
DNAME		VARCHAR(14)
LOC		VARCHAR(13)

## Création d'une Table au Moyen d'une Sous-Interrogation

- Créez une table et insérez des lignes en associant l'ordre CREATE TABLE et l'option *AS subquery*.

```
CREATE TABLE table
      [column(, column...)]
AS subquery;
```

- Le nombre de colonnes spécifiées doit correspondre au nombre de colonnes de la sous-interrogation.
- Définissez des colonnes avec des noms de colonne et des valeurs par défaut.

## Création d'une Table au Moyen d'une Sous-Interrogation

```
SQL> CREATE TABLE dept30
2 AS
3 SELECT empno, ename, sal*12 ANNSAL, hiredate
4 FROM emp
5 WHERE deptno = 30;
Table created.
```

```
SQL> DESCRIBE dept30
```

Name	NULL?	Type
EMPNO		NUMBER(4)
ENAME		VARCHAR2(10)
ANNSAL		NUMBER
HIREDATE		DATE

## L'ordre ALTER TABLE

Utilisez l'ordre ALTER TABLE pour :

- Ajouter une colonne
- Modifier une colonne existante
- Définir une valeur par défaut pour une nouvelle colonne

```
ALTER TABLE table
ADD (column datatype [DEFAULT expr]
[, column datatype]...);
```

```
ALTER TABLE table
MODIFY (column datatype [DEFAULT expr]
[, column datatype]...);
```

## Ajout de Colonnes

**DEPT30**

EMPNO	ENAME	ANNSAL	HIREDATE	<b>Nouvelle colonne</b>
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	
...				

**DEPT30**

EMPNO	ENAME	ANNSAL	HIREDATE	JOB
7698	BLAKE	34200	01-MAY-81	
7654	MARTIN	15000	28-SEP-81	
7499	ALLEN	19200	20-FEB-81	
7844	TURNER	18000	08-SEP-81	
...				

**"...ajouter une nouvelle colonne à la table DEPT30..."**

## Ajout de Colonnes

- Utilisez la clause **ADD** pour ajouter des colonnes.

```
SQL> ALTER TABLE dept30
2 ADD (job VARCHAR2(9));
Table altered.
```

- La nouvelle colonne est placée à la fin.

```
EMPNO ENAME ANNSAL HIREDATE JOB
-----
7698 BLAKE 34200 01-MAY-81
7654 MARTIN 15000 28-SEP-81
7499 ALLEN 19200 20-FEB-81
7844 TURNER 18000 08-SEP-81
...
6 rows selected.
```



## Modification de Colonnes

- Vous pouvez modifier le type de données, la taille et la valeur par défaut d'une colonne.

```
ALTER TABLE dept30  
MODIFY (ename VARCHAR2(15));  
Table altered.
```

- La modification d'une valeur par défaut ne s'applique qu'aux insertions ultérieures dans la table.

## Suppression de Tables

- La structure et toutes les données de la table sont supprimées.
- Une suppression de table ne peut être annulée.

```
SQL> DROP TABLE dept30;  
Table dropped.
```

# Les Contraintes

ORACLE®

## Objectifs

**A la fin de ce chapitre, vous saurez :**

- Définir les contraintes
- Créer des contraintes et les maintenir

ORACLE®

## Les Contraintes

- Les contraintes contrôlent des règles de gestion au niveau d'une table.
- Les contraintes empêchent la suppression d'une table lorsqu'il existe des dépendances.
- Types de contraintes valides dans Oracle :
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK

ORACLE®

## Conventions Applicables aux Contraintes

- Si vous ne nommez pas une contrainte, Oracle8 Server créera un nom au format SYS\_Cn.
- Vous pouvez créer une contrainte :
  - En même temps que la création de la table
  - Une fois que la table est créée
- Définissez une contrainte peut être définie au niveau table ou colonne.
- Consulter le dictionnaire de données pour retrouver une contrainte.

ORACLE®

## Les Contraintes

```
CREATE TABLE [schema.] table
  (column datatype [DEFAULT expr]
   [column_constraint],
   ...
   [table_constraint]);
```

```
CREATE TABLE emp(
  empno  NUMBER(4),
  ename  VARCHAR2(10),
  ...
  deptno NUMBER(2) NOT NULL,
  CONSTRAINT emp_empno_pk
           PRIMARY KEY (EMPNO));
```

ORACLE®

## Les Contraintes

- **Contrainte au niveau colonne**

```
column [CONSTRAINT constraint_name] constraint_type,
```

- **Contrainte au niveau table**

```
column,...
  [CONSTRAINT constraint_name] constraint_type
  (column, ...),
```

ORACLE®

## La Contrainte NOT NULL

Interdit la présence de valeurs NULL dans la colonne

EMP

EMPNO	ENAME	JOB	...	COMM	DEPTNO
7839	KING	PRESIDENT			10
7698	BLAKE	MANAGER			30
7782	CLARK	MANAGER			10
7566	JONES	MANAGER			20
...					

↑  
Contrainte NOT NULL  
(aucune ligne ne peut  
avoir de valeur NULL  
dans cette colonne)

↑  
Absence de contrainte  
NOT NULL  
(toute ligne peut avoir  
une valeur NULL dans  
cette colonne)

↑  
Contrainte  
NOT NULL

ORACLE®

## La Contrainte NOT NULL

Se définit au niveau colonne

```
SQL> CREATE TABLE emp (  
2     empno    NUMBER(4) ,  
3     ename    VARCHAR2(10) NOT NULL ,  
4     job      VARCHAR2(9) ,  
5     mgr      NUMBER(4) ,  
6     hiredate DATE ,  
7     sal      NUMBER(7,2) ,  
8     comm     NUMBER(7,2) ,  
9     deptno   NUMBER(2) NOT NULL) ;
```

ORACLE®

## La Contrainte de Clé UNIQUE

contrainte de clé UNIQUE

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

A insérer

50	SALES	DETROIT
60		BOSTON

Interdit (DNAME=SALES existe déjà)

Autorisé

ORACLE®

## La Contrainte de Clé UNIQUE

Se définit au niveau table ou colonne

```
SQL> CREATE TABLE dept (  
2     deptno    NUMBER(2) ,  
3     dname     VARCHAR2(14) ,  
4     loc       VARCHAR2(13) ,  
5     CONSTRAINT dept_dname_uk UNIQUE(dname) );
```

ORACLE®

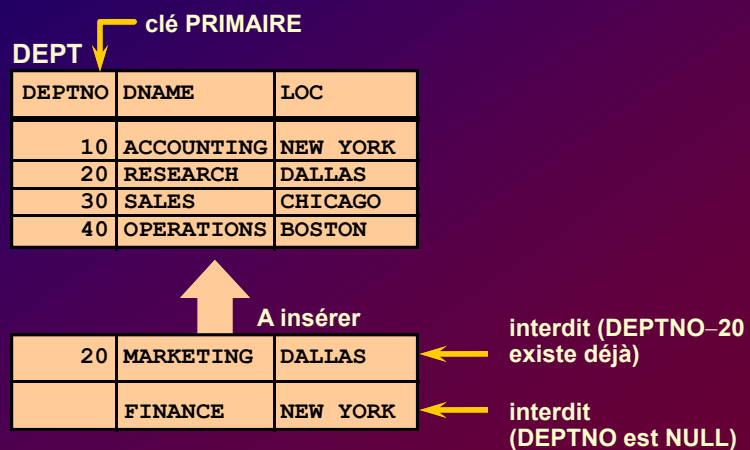
## La Contrainte CHECK

- Définit une condition que chaque ligne doit obligatoirement satisfaire

```
..., deptno NUMBER(2),  
    CONSTRAINT emp_deptno_ck  
        CHECK (DEPTNO BETWEEN 10 AND 99),...
```

ORACLE®

## La Contrainte PRIMARY KEY



ORACLE®

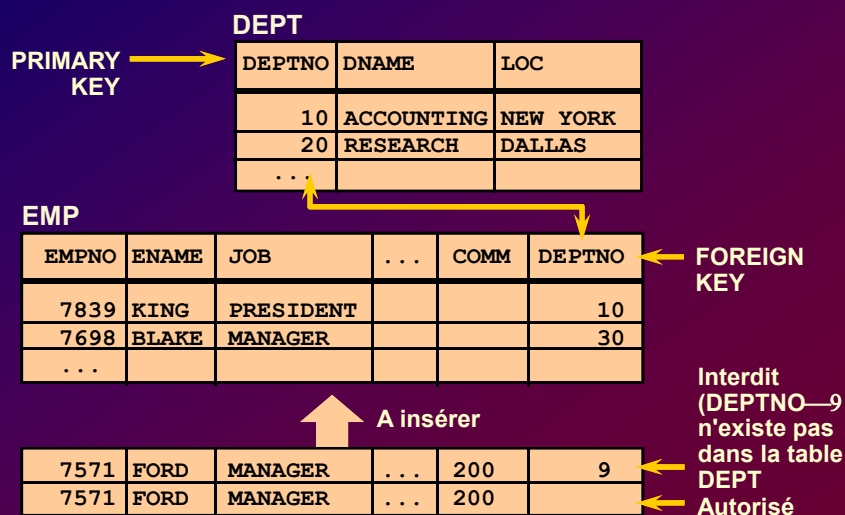
## La Contrainte PRIMARY KEY

Se définit au niveau table ou colonne

```
SQL> CREATE TABLE dept (
2     deptno    NUMBER(2) ,
3     dname     VARCHAR2(14) ,
4     loc       VARCHAR2(13) ,
5     CONSTRAINT dept_dname_uk UNIQUE (dname) ,
6     CONSTRAINT dept_deptno_pk PRIMARY KEY(deptno)) ;
```

ORACLE®

## La Contrainte FOREIGN KEY



ORACLE®



## La Contrainte FOREIGN KEY

Se définit au niveau table ou colonne

```
SQL> CREATE TABLE emp(  
2     empno      NUMBER(4),  
3     ename      VARCHAR2(10) NOT NULL,  
4     job        VARCHAR2(9),  
5     mgr        NUMBER(4),  
6     hiredate   DATE,  
7     sal        NUMBER(7,2),  
8     comm       NUMBER(7,2),  
9     deptno     NUMBER(2) NOT NULL,  
10    CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno)  
11           REFERENCES dept (deptno));
```

ORACLE®

## Mots-clés Associés à la Contrainte FOREIGN KEY

- **FOREIGN KEY**
  - Définit la colonne dans la table détail dans une contrainte de niveau table
- **REFERENCES**
  - Identifie la table et la colonne de la table maître
- **ON DELETE CASCADE**
  - Autorise la suppression d'une ligne dans la table maître et des lignes dépendantes dans la table détail

ORACLE®

## Ajout d'une Contrainte

```
ALTER TABLE table
ADD [CONSTRAINT constraint] type (column);
```

- Vous pouvez ajouter ou supprimer une contrainte, mais pas la modifier
- Vous pouvez activer ou désactiver des contraintes
- Pour ajouter une contrainte NOT NULL, utilisez la clause MODIFY

ORACLE®

## Ajout d'une Contrainte

Ajouter une contrainte FOREIGN KEY à la table EMP précisant qu'un manager doit déjà exister dans la table EMP en tant qu'employé valide.

```
SQL> ALTER TABLE      emp
   2  ADD CONSTRAINT emp_mgr_fk
   3          FOREIGN KEY (mgr) REFERENCES emp (empno);
Table altered.
```

ORACLE®

## Suppression d'une Contrainte

- Supprimer de la table EMP la contrainte concernant le manager.

```
SQL> ALTER TABLE      emp
      2 DROP CONSTRAINT  emp_mgr_fk;
Table altered.
```

- Supprimer la contrainte PRIMARY KEY de la table DEPT, ainsi que la contrainte FOREIGN KEY associée définie sur la colonne EMP.DEPTNO.

```
SQL> ALTER TABLE      dept
      2 DROP PRIMARY KEY CASCADE;
Table altered.
```

ORACLE®

## Désactivation de Contraintes

- Pour désactiver une contrainte d'intégrité, utiliser la clause DISABLE de l'ordre ALTER TABLE.
- Pour désactiver les contraintes d'intégrité dépendantes, ajouter l'option CASCADE.

```
SQL> ALTER TABLE      emp
      2 DISABLE CONSTRAINT emp_empno_pk CASCADE;
Table altered.
```

ORACLE®

## Activation de Contraintes

- Pour activer une contrainte d'intégrité actuellement désactivée dans la définition de la table, utiliser la clause **ENABLE**

```
SQL> ALTER TABLE      emp
      2  ENABLE CONSTRAINT emp_empno_pk;
Table altered.
```

- Si vous activez une contrainte **UNIQUE** ou **PRIMARY KEY**, un index correspondant est automatiquement créé.

ORACLE®

## Vérification des Contraintes

Pour afficher les définitions et noms de toutes les contraintes, interrogez la table **USER\_CONSTRAINTS**.

```
SQL> SELECT constraint_name, constraint_type,
      2      search_condition
      3  FROM   user_constraints
      4  WHERE  table_name = 'EMP';
```

CONSTRAINT_NAME	C SEARCH_CONDITION
SYS_C00674	C EMPNO IS NOT NULL
SYS_C00675	C DEPTNO IS NOT NULL
EMP_EMPNO_PK	P
...	

ORACLE®

## Résumé

- Vous pouvez créer des contraintes de type :
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK
- Utilisez la table **USER\_CONSTRAINTS** pour afficher les noms et définitions de toutes les contraintes.

ORACLE®

Pour la gestion d'un comptoir commercial, nous proposons le schéma relationnel ci-dessous :

- **CLIENT** (codeclt, nomclt, prenomclt, adresse, cp, ville)
- **COMMANDE** (numero, date\_c, total, codeclt, reference)
- **PRODUIT** (reference, nomPro, prix, codeCat)
- **CATEGORIE** (codeCat, nomCat, description)

A. Créez ce modèle en proposant les contraintes et les types de données appropriés ?

B. Insérez quelques lignes dans chaque table, puis testez avec des requêtes de sélection le contenu de chaque table ?

ORACLE®