# Tables Utilisées dans le Cours

**EMP**

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|---|---|---|---|---|---|---|---|
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 1500 | | 10 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7788 | SCOTT | ANALYST | 7566 | 09-DEC-82 | 3000 | | 20 |
| 7876 | ADAMS | CLERK | 7788 | 12-JAN-83 | 1100 | | 20 |
| | | | 7782 | 23-JAN-82 | | | 10 |

**DEPT**

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

**SALGRADE**

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

## Tables Utilisées dans le Cours

Vous utiliserez principalement trois tables dans ce cours :

- La table EMP qui contient des informations sur tous les employés
- La table DEPT, qui contient des informations sur tous les départements
- La table SALGRADE , contenant des informations sur les différents niveaux de salaires en fonction de l'échelon

La structure et les données de chaque table sont données dans l'annexe B.

## Exercices 1

1. Initialisez une session SQL*Plus avec votre ID et le mot de passe que votre instructeur vous a remis.
2. Les commandes SQL*Plus accèdent aux bases de données.
   Vrai/Faux
3. L'ordre SELECT suivant sera convenablement exécuté.
   Vrai/Faux

```
SQL> SELECT       ename, job, sal Salary
  2  FROM         emp;
```

4. L'ordre SELECT suivant sera convenablement exécuté.
   Vrai/Faux

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

5. Cet ordre comporte trois erreurs de code ; pouvez-vous les trouver ?

```
SQL> SELECT       empno, ename
  2               salary x 12 ANNUAL SALARY
  3  FROM         emp;
```

6. Affichez la structure de la table DEPT. Sélectionnez toutes les données de la table DEPT.

```
Name          Null?    Type
------------ -------- ----
DEPTNO        NOT NULL NUMBER(2)
DNAME                  VARCHAR2(14)
LOC                    VARCHAR2(13)
```

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

## Exercices 1

7. Affichez la structure de la table EMP . Créez une requête pour afficher le nom (ename), le poste (job), la date d'embauche (hiredate) et le matricule (empno) de chaque employé, en plaçant le matricule en premier. Enregistrez votre ordre SQL dans un fichier nommé *p1q7.sql*.

```
Name                      Null?    Type
------------------------- -------- ----
EMPNO                     NOT NULL NUMBER(4)
ENAME                              VARCHAR2(10)
JOB                                VARCHAR2(9)
MGR                                NUMBER(4)
HIREDATE                           DATE
SAL                                NUMBER(7,2)
COMM                               NUMBER(7,2)
DEPTNO                             NOT NULL NUMBER(2)
```

8. Exécutez la requête que vous avez placée dans le fichier *p1q7.sql*.

```
EMPNO ENAME      JOB              HIREDATE
----- ---------- ---------        ---------
 7839 KING       PRESIDENT        17-NOV-81
 7698 BLAKE      MANAGER          01-MAY-81
 7782 CLARK      MANAGER          09-JUN-81
 7566 JONES      MANAGER          02-APR-81
 7654 MARTIN     SALESMAN         28-SEP-81
 7499 ALLEN      SALESMAN         20-FEB-81
 7844 TURNER     SALESMAN         08-SEP-81
 7900 JAMES      CLERK            03-DEC-81
 7521 WARD       SALESMAN         22-FEB-81
 7902 FORD       ANALYST          03-DEC-81
 7369 SMITH      CLERK            17-DEC-80
 7788 SCOTT      ANALYST          09-DEC-82
 7876 ADAMS      CLERK            12-JAN-83
 7934 MILLER     CLERK            23-JAN-82
14 rows selected.
```

9. Créez une requête pour afficher les différents types de poste existant dans la table EMP.

```
JOB
---------
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN
```

Si vous avez du temps, faites les exercices suivants :

10. Editez *p1q7.sql*. Donnez respectivement les noms suivants aux en-têtes de colonne : Emp #, Employee, Job, et Hire Date. Exécutez à nouveau votre requête.

```
Emp # Employee  Job         Hire Date
----- --------- ---------- ---------------
 7839 KING      PRESIDENT  17-NOV-81
 7698 BLAKE     MANAGER    01-MAY-81
 7782 CLARK     MANAGER    09-JUN-81
 7566 JONES     MANAGER    02-APR-81
 7654 MARTIN    SALESMAN   28-SEP-81
 7499 ALLEN     SALESMAN   20-FEB-81
 7844 TURNER    SALESMAN   08-SEP-81
 7900 JAMES     CLERK      03-DEC-81
 7521 WARD      SALESMAN   22-FEB-81
 7902 FORD      ANALYST    03-DEC-81
 7369 SMITH     CLERK      17-DEC-80
 7788 SCOTT     ANALYST    09-DEC-82
 7876 ADAMS     CLERK      12-JAN-83
 7934 MILLER    CLERK      23-JAN-82
14 rows selected.
```

## Exercices 1

11. Affichez le nom concaténé avec le poste en les séparant par une virgule suivie d'un espace, puis donnez comme titre à la colonne Employee and Title.

```
Employee and Title
------------------
KING, PRESIDENT
BLAKE, MANAGER
CLARK, MANAGER
JONES, MANAGER
MARTIN, SALESMAN
ALLEN, SALESMAN
TURNER, SALESMAN
JAMES, CLERK
WARD, SALESMAN
FORD, ANALYST
SMITH, CLERK
SCOTT, ANALYST
ADAMS, CLERK
MILLER, CLERK
14 rows selected.
```

Si vous souhaitez aller plus loin dans la difficulté, faites l' exercice suivant :

12. Créez une requête pour afficher toutes les données de la table EMP dans une seule colonne d'affichage. Séparez chaque colonne par une virgule. Nommez la colonne d'affichage THE_OUTPUT.

```
THE_OUTPUT
--------------------------------------------------
7839,KING,PRESIDENT,,17-NOV-81,5000,,10
7698,BLAKE,MANAGER,7839,01-MAY-81,2850,,30
7782,CLARK,MANAGER,7839,09-JUN-81,2450,,10
7566,JONES,MANAGER,7839,02-APR-81,2975,,20
7654,MARTIN,SALESMAN,7698,28-SEP-81,1250,1400,30
7499,ALLEN,SALESMAN,7698,20-FEB-81,1600,300,30
7844,TURNER,SALESMAN,7698,08-SEP-81,1500,0,30
7900,JAMES,CLERK,7698,03-DEC-81,950,,30
7521,WARD,SALESMAN,7698,22-FEB-81,1250,500,30
7902,FORD,ANALYST,7566,03-DEC-81,3000,,20
7369,SMITH,CLERK,7902,17-DEC-80,800,,20
7788,SCOTT,ANALYST,7566,09-DEC-82,3000,,20
7876,ADAMS,CLERK,7788,12-JAN-83,1100,,20
7934,MILLER,CLERK,7782,23-JAN-82,1300,,10
14 rows selected.
```

# Présentation des Exercices

- Interrogation de données et modification de l'ordre des lignes affichées
- Restriction des lignes avec la clause WHERE
- Utilisation des guillemets dans les alias de colonne



## Présentation des Exercices

Vous allez effectuer différents exercices faisant appel aux clauses WHERE et ORDER BY.

## Exercices 2

1. Créez une requête destinee à afficher le nom et le salaire des employes gagnant plus de $2850.
Enregistrez l'ordre SQL créé dans un fichier appele *p2q1.sql*. Exécutez votre requête.

```
ENAME      SAL
---------- ----
KING       5000
JONES      2975
FORD       3000
SCOTT      3000
```

2. Créez une requête destinée à afficher le nom et le numéro de département de l'employé dont le matricule est 7566.

```
ENAME  DEPTNO
------ ------
JONES      20
```

3. Modifiez *p2q1.sql* de manière à afficher le nom et le salaire de tous les employés dont le salaire n'est pas compris entre $1500 et $2850. Enregistrez ce nouvel ordre SQL dans un fichier appelé *p2q3.sql*. Exécutez cette requête.

```
ENAME      SAL
---------- ----
KING       5000
JONES      2975
MARTIN     1250
JAMES       950
WARD       1250
FORD       3000
SMITH       800
SCOTT      3000
ADAMS      1100
MILLER     1300
10 rows selected.
```

4. Affichez le nom, le poste et la date d'entrée (hiredate) des employés embauchés entre le 20 février 1981 et le 1 mai 1981. Classez le résultat par date d'embauche croissante.

```
ENAME      JOB       HIREDATE
---------- --------- -----------
ALLEN      SALESMAN  20-FEB-81
WARD       SALESMAN  22-FEB-81
JONES      MANAGER   02-APR-81
BLAKE      MANAGER   01-MAY-81
```

5. Affichez le nom et le numéro de département de tous les employés des départements 10 et 30 classés par ordre alphabétique des noms.

```
ENAME          DEPTNO
---------- ---------
ALLEN              30
BLAKE              30
CLARK              10
JAMES              30
KING               10
MARTIN             30
MILLER             10
TURNER             30
WARD               30
9 rows selected.
```

6. Modifiez *p2q3.sql* pour afficher la liste des noms et salaires des employés gagnant plus de $1500 et travaillant dans le département 10 ou 30. Nommez les colonnes Employee et Monthly Salary, respectivement. Enregistrez à nouveau votre ordre SQL dans un fichier appelé *p2q6.sql*. Réexécutez votre requête.

```
Employee  Monthly Salary
--------  --------------
KING                5000
BLAKE               2850
CLARK               2450
ALLEN               1600
```

## Exercices 2

7. Affichez le nom et la date d'embauche de chaque employé entré en 1982.

```
ENAME  HIREDATE
------ ---------
SCOTT  09-DEC-82
MILLER 23-JAN-82
```

8. Affichez le nom et le poste de tous les employés n'ayant pas de manager.

```
ENAME JOB
----- ---------
KING  PRESIDENT
```

9. Affichez le nom, le salaire et la commission de tous les employés qui perçoivent des commissions. Triez les données dans l'ordre décroissant des salaires et des commissions.

```
ENAME    SAL    COMM
------- ----  ----
ALLEN    1600    300
TURNER   1500      0
MARTIN   1250   1400
WARD     1250    500
```

Si vous avez le temps, faites les exercices suivants :

10. Affichez le nom de tous les employés dont la troisième lettre du nom est un *A*.

```
ENAME
----------
BLAKE
CLARK
ADAMS
```

11. Affichez le nom de tous les employés dont le nom contient deux *L* et travaillant dans le département 30 ou dont le manager est 7782.

```
ENAME
----------
ALLEN
MILLER
```

## Exercices 2

Si vous voulez aller plus loin dans la difficulté, faites les exercices suivants :

12. Affichez le nom, le poste et le salaire de tous les ' CLERK' ou 'ANALYST' dont le salaire est différent de $1000, $3000 ou $5000.

```
ENAME  JOB          SAL
------ -------- -- ---
JAMES  CLERK         950
SMITH  CLERK         800
ADAMS  CLERK        1100
MILLER CLERK        1300
```

13. Afficher le nom, le salaire et la commission de tous les employés dont le montant de commission est de plus de 10% supérieur au salaire. Enregistrez votre requête sous le nom *p2q13.sql*.

```
ENAME      SAL   COMM
. ------ ------ -- - -
MARTIN     1250   1400
```

# Fonctions Mono-Ligne

# Objectifs

A la fin de ce chapitre, vous saurez :

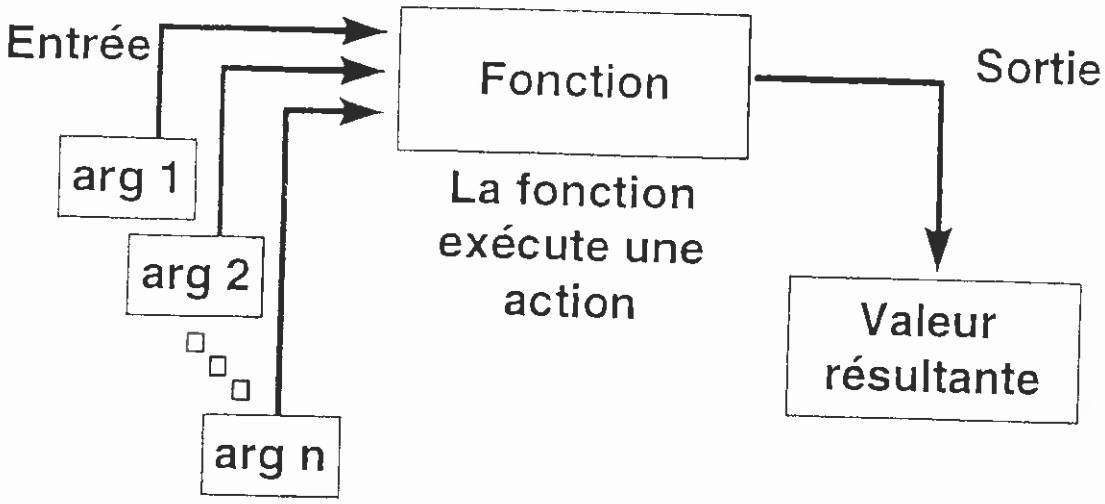
- Décrire différents types de fonctions SQL
- Utiliser les fonctions caractère, numériques et date dans les ordres SELECT
- Expliquer les fonctions de conversion



Objectifs

Les fonctions rendent l'instruction SELECT plus puissante en permettant de manipuler des valeurs de données. Ce chapitre est le premier de deux consacrés aux fonctions. Il explique en particulier les fonctions caractère, numériques et date, ainsi que les fonctions qui convertissent des données d'un certain type en un autre type, par exemple, des données de type caractère en données numériques.

# Fonctions SQL




3-3

## Fonctions SQL

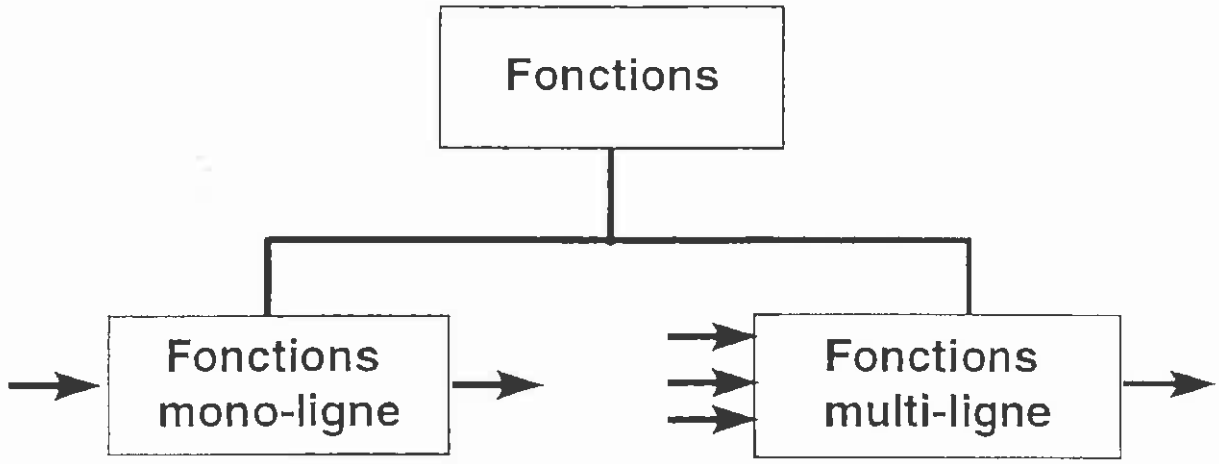
Les fonctions représentent une caractéristique très puissant de SQL et sont utilisées pour :

- Effectuer des calculs sur des données
- Transformer des données
- Effectuer des calculs sur des groupes de lignes
- Formater des dates et des nombres pour l'affichage
- Convertir des types de données de colonnes

Les fonctions SQL acceptent les arguments et ramènent des valeurs.

**Remarque:** la plupart des fonctions SQL décrites dans ce chapitre sont spécifiques au SQL d'Oracle.

# Deux Types de Fonctions SQL




3-4

## Fonctions SQL

Il existe deux types de fonctions :

- Les fonctions mono-ligne
- Les fonctions multi-ligne

### Fonctions Mono-Ligne

Ces fonctions agissent sur une seule ligne à la fois et ramènent un seul résultat. Il existe plusieurs types de fonctions mono-ligne. Ce chapitre décrit les quatre suivantes :

- Caractère
- Numérique
- Date
- Conversion

### Fonctions Multi-Ligne

Ces fonctions manipulent des groupes de lignes et ramènent un seul résultat par groupe de lignes.

Pour obtenir la syntaxe et la liste complète des fonctions disponibles, reportez-vous à *Oracle 8 Server SQL Reference, Release 8*

# 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

```
function_name (column|expression, [arg1, arg2,...])
```

3-5

## Fonctions Mono-Ligne

Les fonctions mono-ligne sont utilisées pour manipuler des éléments de données. Elles acceptent un ou plusieurs arguments et ramènent une seule valeur par ligne issue de la requête. Un argument peut être l'un des éléments suivants :

- Une valeur constante utilisateur
- Une variable
- Un nom de colonne
- Une expression

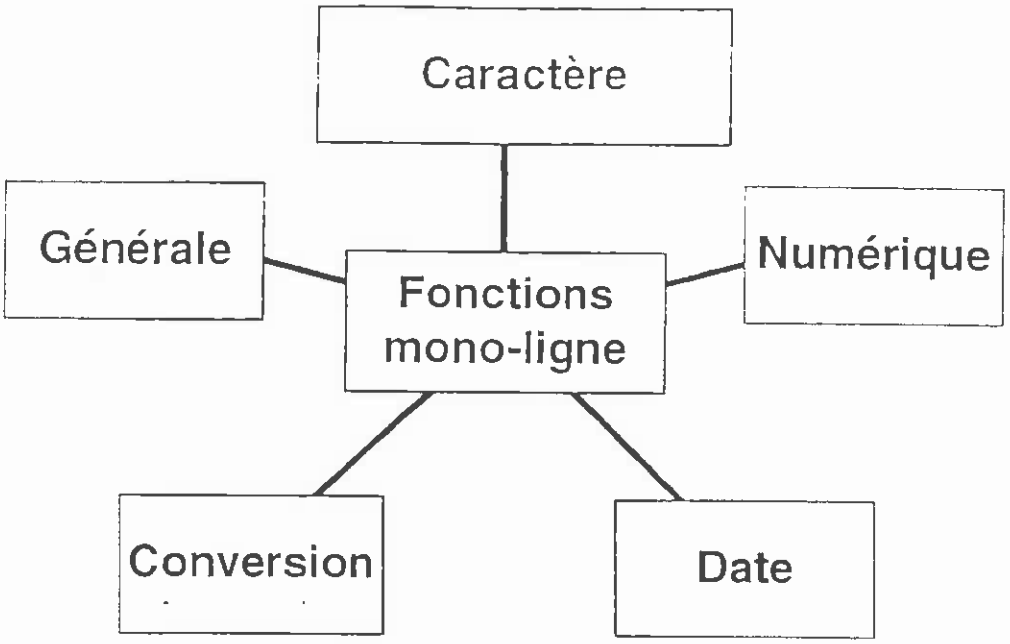
## Caractéristiques des Fonctions Mono-Ligne

- Acceptent un ou plusieurs arguments.
- Agissent sur chacune des lignes issues de la requête.
- Ramènent un résultat par ligne.
- Peuvent ramener une valeur d'un type différent de celui mentionné.
- S'utilisent dans les clauses SELECT, WHERE, et ORDER BY.
- Peuvent être imbriquées.

Syntaxe :

| | |
|---|---|
| *function_name* | nom de la fonction |
| *column* | nom d'une colonne de la base de données |
| *expression* | chaîne de caractères ou expression calculée |
| *arg1, arg2* | argument utilisé dans la fonction |

# Fonctions Mono-Ligne



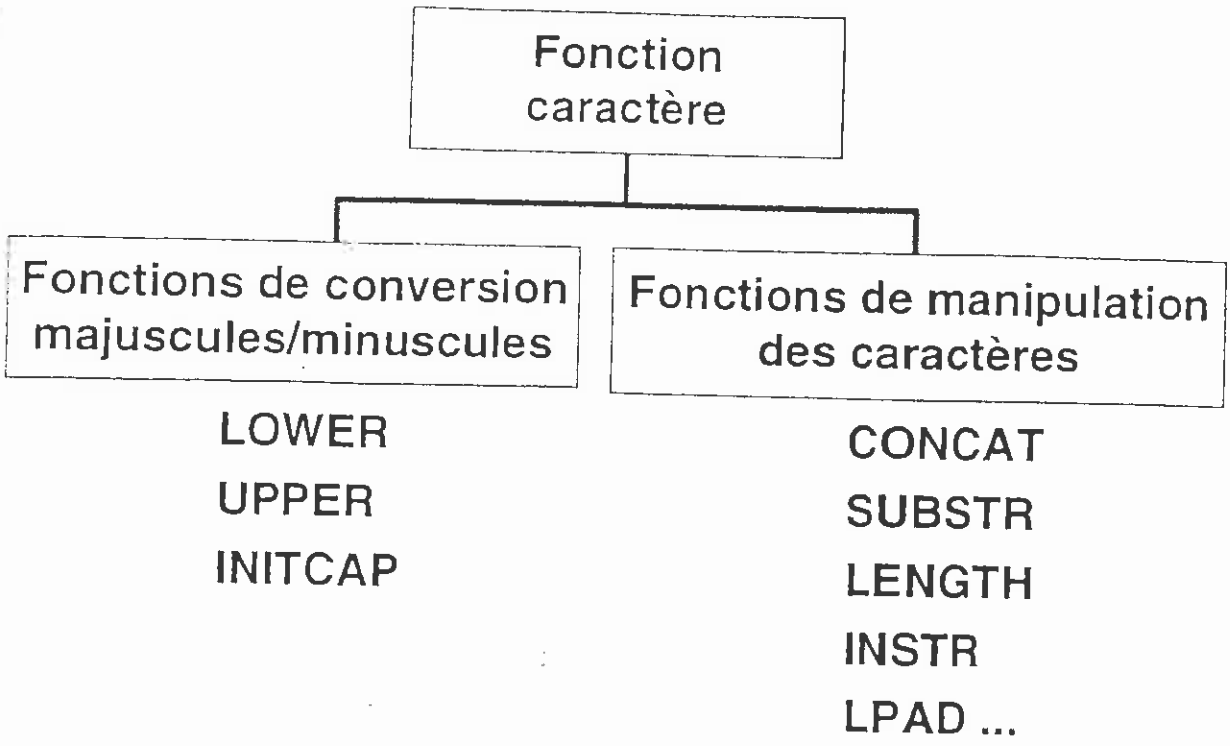

3-6

## Fonctions Mono-Ligne

Ce chapitre présente les fonctions mono-ligne suivantes :

- Fonctions caractère : elles acceptent des caractères en entrée et peuvent ramener des valeurs caractère ou numériques.
- Fonctions numériques : elles acceptent des nombres en entrée et ramènent des valeurs numériques.
- Fonctions date : elles opèrent sur des valeurs de type date et ramènent des valeurs de type date. Seule la fonction MONTHS_BETWEEN ramène une valeur numérique.
- Fonctions de conversion : elles convertissent une valeur d'un certain type dans un autre type
- Fonctions générales
  - fonction NVL
  - fonction DECODE

# Fonctions Caractère




## Fonctions caractère

Les fonctions mono-ligne caractère acceptent des données caractère en entrée et ramènent des données caractère ou numériques. Les fonctions caractère se divisent en deux groupes :

- Les fonctions de conversion majuscules/minuscules
- Les fonctions de manipulation des caractères

| Fonction | Modification |
|---|---|
| LOWER(*columnlexpression*) | Convertit les caractères alphabétiques en minuscules. |
| UPPER(*columnlexpression*) | Convertit les caractères alphabétiques en majuscules. |
| INITCAP(*columnlexpression*) | Convertit l'initiale de chaque mot en majuscule et les caractères suivants en minuscules. |
| CONCAT(*column1expression1, column2lexpression2*) | Concatène la première chaîne de caractère à la seconde. Equivaut à l'opérateur de concaténation (\|\|). |
| SUBSTR(*columnlexpression,m[,n]*) | Extrait une partie de la chaîne de caractères en commençant au caractère situé à la position *m* et sur une longueur de *n* caractères.<br>Si *m* est une valeur négative, le décompte s'effectue dans le sens inverse (à partir du dernier caractère de la chaîne).<br>Si *n* est omis, tous les caractères jusqu'à la fin de la chaîne sont ramenés. |
| LENGTH(*columnlexpression*) | Ramène le nombre de caractères d'une chaîne de caractères. |
| INSTR(*columnlexpression, m*) | Ramène une valeur égale à la position du caractère *m*. |
| LPAD(*columnlexpression, n, 'string'*) | Complète une chaîne de caractère sur la gauche avec la chaîne *'string'* pour parvenir à une longueur totale de *n* caractères. |

**Remarque :** cette liste de fonctions caractère n'est pas complète :

Pour plus d'informations, reportez-vous à
*Oracle8 Server SQL Reference, Release 8*, "Character Functions."

# Fonctions de Conversion Majuscules/Minuscules

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

3-8

## Fonctions de Conversion Majuscules/Minuscules

LOWER, UPPER, et INITCAP sont les trois fonctions qui modifient la casse.

- LOWER : Convertit tous les caractères d'une chaîne en minuscules
- UPPER : Convertit tous les caractères d'une chaîne en majuscules
- INITCAP : Convertit la première lettre de chaque mot en majuscule et les lettres suivantes en minuscules

```
SQL> SELECT  'The job title for '||INITCAP(ename)||' is '
  2                 ||LOWER(job) AS "EMPLOYEE DETAILS"
  3  FROM     emp;
```

```
EMPLOYEE DETAILS
-----------------------------------------
The job title for King is president
The job title for Blake is manager
The job title for Clark is manager
...
14 rows selected.
```

## 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
```

3-9

### Fonctions de Conversion Majuscules/Minuscules

L'exemple ci-dessus affiche le matricule, le nom et le numéro de département de l'employé BLAKE.

La clause WHERE du premier ordre SQL spécifie le nom de l'employé sous la forme 'blake'. Comme les données de la table EMP sont stockées en majuscules, il est impossible de trouver le nom correspondant dans la table EMP, et par conséquent de sélectionner des lignes.

La clause WHERE du second ordre SQL indique que le nom d'employé dans la table EMP doit être converti en minuscules pour être ensuite comparé à 'blake'. Les deux noms étant maintenant en minuscules, la ligne correspondante est ramenée. La clause WHERE, réécrite de la manière suivante, produit le même résultat :

```
... WHERE   ename = 'BLAKE'
```

Le nom de l'employé apparaît à l'affichage tel qu'il est stocké dans la base de données. Pour obtenir le nom avec seulement l'initiale en majuscule, il suffit d'utiliser la fonction INITCAP dans l'ordre SELECT.

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

# 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 |

3-10

### Fonctions de Manipulation des Caractères

CONCAT, SUBSTR, LENGTH, INSTR et LPAD sont les cinq fonctions de manipulation des caractères étudiées dans ce chapitre.

- CONCAT : Concatène des valeurs. Le nombre de paramètres avec CONCAT est limité à deux.
- SUBSTR : Extrait une chaîne de longueur déterminée.
- LENGTH : Fournit la valeur numérique correspondant au nombre de caractères d'une chaîne.
- INSTR : Fournit la valeur numérique correspondant à la position d'un caractère.
- LPAD : Ajoute des caractères de remplissage à la gauche d'une valeur alphanumérique qui sera ainsi cadrée à droite.

**Remarque :** la fonction de manipulation des caractères RPAD ajoute des caractères de remplissage à la droite d'une valeur alphanumérique qui sera ainsi cadrée à gauche.

# 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 |

3-11

## Fonctions de Manipulation des Caractères

L'exemple ci-dessus affiche le nom des employés concaténé à leur poste, le nombre de caractères du nom, ainsi que la position de la lettre A dans leur nom et ce, pour tous les employés dont le poste commence par la chaîne 'SALES'.

**Exemple**

Modifier l'ordre SQL ci-dessus afin d'afficher les données concernant les employés dont le nom se termine par un N.

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

| ENAME | CONCAT(ENAME,JOB) | LENGTH(ENAME) | INSTR(ENAME,'A') |
|---|---|---|---|
| MARTIN | MARTINSALESMAN | 6 | 2 |
| ALLEN | ALLENSALESMAN | 5 | 1 |

## Fonctions Numériques

- ROUND : Arrondit la valeur à la précision spécifiée

  ROUND(45.926, 2) ⟶ 45.93

- TRUNC : Tronque la valeur à la précision spécifiée

- TRUNC(45.926, 2) ⟶ 45.92

- MOD : Ramène le reste d'une division

  MOD(1600, 300) ⟶ 100

3-12

### Fonctions numériques

Les fonctions numériques utilisent et ramènent des valeurs numériques. Cette section décrit quelques-unes de ces fonctions.

| Fonction | Modification |
|---|---|
| ROUND(*columnlexpression, n)* | Arrondit la valeur de la colonne ou de l'expression à une précision de $10^{-n}$.<br>Si *n* est positif, le nombre sera arrondi à *n* décimales.<br>Si *n* est omis, il n'y aura pas de décimale.<br>Si *n* est négatif, l'arrondi portera sur la partie du nombre située à gauche de la virgule (dizaine, centaine...) |
| TRUNC(*columnlexpression,n*) | Tronque la valeur de la colonne ou de l'expression à une précision de $10^{-n}$.<br>Si *n* est positif, le nombre sera tronqué à *n* décimales.<br>Si *n* est omis, il n'y aura pas de décimale.<br>Si *n* est négatif, ce sera la partie du nombre située à gauche de la virgule (dizaine, centaine...) qui sera tronquée. |
| MOD(*m,n*) | Ramène le reste de la division de *m* par *n*. |

**Remarque :** Cette liste de fonctions numériques n'est pas exhaustive.

Pour plus d'informations, reportez-vous à
*Oracle8 Server SQL Reference, Release 8,* "Number Functions."

# Utilisation de la Fonction ROUND

## Affichage de la valeur 45.923 arrondie au centième, à 0 décimale et à la dizaine supérieure.

```
SQL> SELECT ROUND(45.923,2), ROUND(45.923,0),
  2         ROUND(45.923,-1)
  3  FROM   DUAL;
```

```
ROUND(45.923,2) ROUND(45.923,0) ROUND(45.923,-1)
--------------- --------------- ----------------
          45.92              46               50
```

3-13

### Fonction ROUND

La fonction ROUND arrondit la valeur d'une colonne ou d'une expression à une précision égale à $10^{-n}$. Lorsque $n$ vaut 0 ou est absent, la valeur est arrondie à zéro décimale. Si $n$ vaut 2, la valeur est arrondie à deux décimales (soit au centième). Inversement, si $n$ vaut -2, la valeur est arrondie de deux chiffres à gauche de la virgule, soit à la centaine.

La fonction ROUND peut aussi être utilisée avec les fonctions date. Nous verrons quelques exemples un peu plus loin dans ce chapitre.

La table DUAL est une table factice. Nous y reviendrons ultérieurement.

# Utilisation de la Fonction TRUNC

## Affichage de la valeur 45.923 tronquée au centième, à 0 décimale et à la dizaine.

```
SQL> SELECT TRUNC(45.923,2), TRUNC(45.923),
  2         TRUNC(45.923,-1)
  3  FROM   DUAL;
```

```
TRUNC(45.923,2) TRUNC(45.923) TRUNC(45.923,-1)
--------------- -------------- ----------------
          45.92             45               40
```

3-14

### Fonction TRUNC

La fonction TRUNC tronque la valeur de la colonne ou de l'expression à une précision égale à $10^{-n}$.

La fonction TRUNC fonctionne avec des arguments identiques à ceux de la fonction ROUND. Lorsque *n* vaut 0 ou est absent, la valeur est tronquée à zéro décimale. Si *n* vaut 2, la valeur est tronquée à deux décimales (soit au centième). Inversement, si *n* vaut -2, la valeur est tronquée de deux chiffres à gauche de la virgule, soit à la centaine.

Comme la fonction ROUND, la fonction TRUNC peut aussi être utilisée avec les fonctions date.

# 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 |

3-15

### Fonction MOD

La fonction MOD calcule le reste de la division d'une valeur1 par une valeur2. L'exemple ci-dessus donne le reste de la division du salaire par la commission, pour tous les employés occupant un poste de vendeur (SALESMAN).

# Utilisation des Dates

- Oracle stocke les dates dans un format numérique interne : siècle, année, mois, jour, heures, minutes, secondes.
- Le format de date par défaut est DD-MON-YY.
- La fonction SYSDATE ramène la date et l'heure courante.
- DUAL est une table factice qu'on peut utiliser pour visualiser SYSDATE.

### Format de Date Oracle

Oracle stocke les dates dans un format numérique interne représentant le siècle, l'année, le mois, le jour, les heures, les minutes et les secondes.

Le format d'entrée et d'affichage par défaut des dates est DD-MON-YY. Les dates valides pour Oracle sont comprises entre le 1er janvier 4712 av.J.-C. et le 31 décembre 9999 apr.J.-C.

### SYSDATE

SYSDATE est une fonction date qui permet d'obtenir la date et l'heure courante. SYSDATE peut être utilisé de même façon qu'un nom de colonne quelconque. Il est usuel d'interroger la table "factice" DUAL.

### DUAL

La table DUAL appartient à l'utilisateur SYS, mais tous les utilisateurs peuvent y accéder. Elle contient une seule colonne, DUMMY, et une seule ligne contenant la valeur X. La table DUAL est utile si vous souhaitez renvoyer une valeur une seule fois, par exemple, la valeur d'une constante, d'une pseudo-colonne ou d'une expression qui ne dépend pas d'une table de données utilisateur.

### Exemple

Afficher la date courante à l'aide de la table DUAL.

```
SQL> SELECT SYSDATE
  2  FROM   DUAL;
```

# 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.
- Ajout *d'un nombre d'heures* à une date en divisant le nombre d'heures par 24.

3-17

## Opérations Arithmétiques sur les Dates

Comme la base de donnée stocke les dates en tant que données numériques, il est possible d'effectuer des calculs tels que l'addition ou la soustraction au moyen d'opérateurs arithmétiques. Il est possible d'ajouter et soustraire des constantes numériques aussi bien que des dates.

Les opérations possibles sont les suivantes :

| Opération | Résultat | Description |
|---|---|---|
| date + nombre de jours | Date | Ajoute un certain nombre de jours à une date |
| date - nombre de jours | Date | Soustrait un certain nombre de jours d'une date |
| date - date | Nombre de jours | Soustrait une date d'une autre |
| date + (nombre d'heures)/24 | Date | Ajoute un certain nombre d'heures à une date |

# Utilisation d'Opérateurs Arithmétiques avec les Dates

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

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

3-18

## Opérations Arithmétiques sur les Dates (suite)

L'exemple de la diapositive affiche le nom et le nombre de semaines d'ancienneté de tous les employés du département 10. La date courante (SYSDATE) est soustraite de la date d'embauche de l'employé, puis le résultat est divisé par 7 pour obtenir le nombre de semaines d'ancienneté.

**Remarque :** SYSDATE est une fonction SQL qui donne l'heure et la date courantes. Vos résultats peuvent donc être différents de ceux de l'exemple.

# Fonctions Date

| FONCTION | DESCRIPTION |
|---|---|
| MONTHS_BETWEEN | Nombre de mois situés entre deux dates |
| ADD_MONTHS | Ajoute des mois calendaires à une date |
| NEXT_DAY | Jour qui suit la date spécifiée |
| LAST_DAY | Dernier jour du mois |
| ROUND | Arrondit une date |
| TRUNC | Tronque une date |

3-19

### Fonctions Date

Les fonctions date s'appliquent aux données de type DATE. Toutes les fonctions date ramènent une valeur de type DATE, à l'exception de MONTHS_BETWEEN qui ramène une valeur numérique.

- MONTHS_BETWEEN(*date1*, *date2*) : Donne le nombre de mois situés entre une date (*date1*) et une autre date (*date2*). Le résultat peut être positif ou négatif. Si *date1* est postérieure à *date2*, le résultat est positif ; si *date1* est antérieure à *date2*, le résultat est négatif. La partie non entière du résultat représente une portion de mois.
- ADD_MONTHS(*date*, *n*) : Ajoute un nombre *n* de mois à une *date*. *n* doit être un nombre entier et peut être négatif.
- NEXT_DAY(*date*, '*char*') : Fournit la date de la première occurrence du jour spécifié ('*char*') après la *date* fournie. *char* peut être, soit un numéro de jour de semaine, soit une chaîne de caractères.
- LAST_DAY(*date*) : Indique la date du dernier jour du mois auquel appartient la *date* indiquée.
- ROUND(*date*[,'*fmt*']) : Ramène la *date*, arrondie à l'unité précisée par le modèle de format *fmt*. Lorsque *fmt* est omis, la *date* est arrondie est arrondie au jour le plus proche.
- TRUNC(*date*[, '*fmt*']) : Ramène la *date*, tronquée à l'unité précisée par le modèle de format *fmt*. Lorsque *fmt* est omis, la date est tronquée au jour.

Cette liste des fonctions date n'est pas exhaustive. Les modèles de format sont expliqués dans la suite de ce chapitre. Le mois ou l'année sont des exemples de modèles de format.

# Utilisation des Fonctions Date

- MONTHS_BETWEEN ('01-SEP-95','11-JAN-94') → 19.6774194
- ADD_MONTHS ('11-JAN-94',6) → '11-JUL-94'
- NEXT_DAY ('01-SEP-95','FRIDAY') → '08-SEP-95'
- LAST_DAY('01-SEP-95') → '30-SEP-95'

3-20

## Fonctions Date (suite)

Pour tous les employés ayant moins de 200 mois d'ancienneté, affichez les données suivantes : le matricule, la date d'embauche, le nombre de mois d'ancienneté, la date correspondant à la révision de salaire après 6 mois , le premier vendredi suivant la date d'embauche et le dernier jour du mois d'embauche.

```
SQL> SELECT   empno, hiredate,
  2           MONTHS_BETWEEN(SYSDATE, hiredate) TENURE
  3           ADD_MONTHS(hiredate, 6) REVIEW,
  4           NEXT_DAY(hiredate, 'FRIDAY'), LAST_DAY(hiredate)
  5  FROM     emp
  6  WHERE    MONTHS_BETWEEN (SYSDATE, hiredate)<200;
```

```
    EMPNO HIREDATE     TENURE REVIEW    NEXT_DAY( LAST_DAY(
--------- --------- --------- --------- --------- ---------
     7639 17-NOV-81 192.24794 17-MAY-82 20-NOV-81 30-NOV-81
     7698 01-MAY-81 198.76407 01-NOV-81 08-MAY-81 31-MAY-81
...
11 rows selected.
```

# Utilisation des Fonctions Date

- ROUND('25-JUL-95','MONTH') → 01-AUG-95
- ROUND('25-JUL-95','YEAR') → 01-JAN-96
- TRUNC('25-JUL-95','MONTH') → 01-JUL-95
- TRUNC('25-JUL-95','YEAR') → 01-JAN-95

3-21

### Fonctions Date (suite)

Les fonctions ROUND et TRUNC peuvent être utilisées avec des valeurs de type numérique ou date. Utilisées avec des dates, ces fonctions arrondissent ou tronquent au modèle de format spécifié. Vous pouvez par conséquent arrondir les dates au premier jour du mois ou de l'année les plus proches.

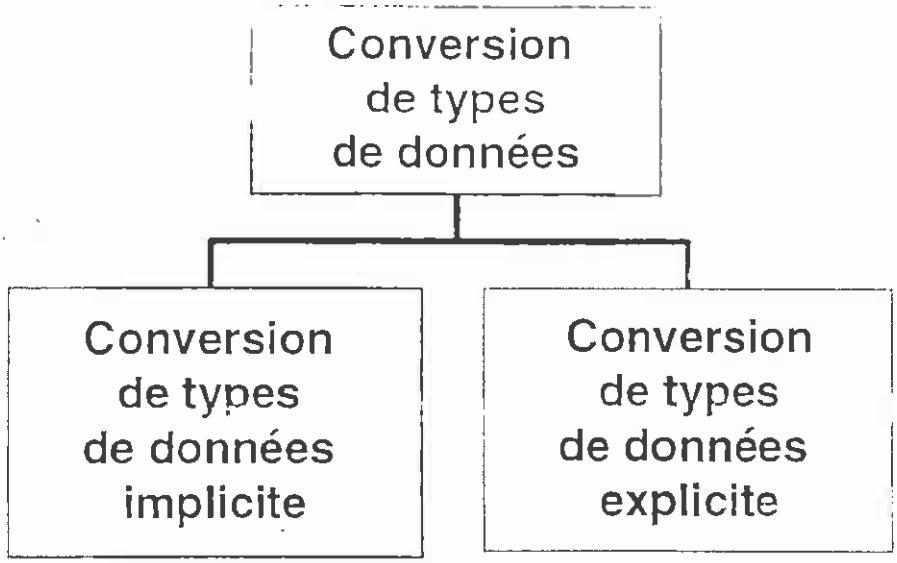
### Exemple

Afficher les dates d'embauche de tous les employés ayant commencé en 1987. Affichez le matricule, la date d'embauche et le mois de début d'activité en utilisant les fonctions ROUND et TRUNC.

```
SQL> SELECT empno, hiredate,
  2         ROUND(hiredate, 'MONTH'), TRUNC(hiredate, 'MONTH')
  3  FROM   emp
  4  WHERE  hiredate like '%87';
```

```
     EMPNO HIREDATE  ROUND(HIR TRUNC(HIR
---------- --------- --------- ---------
      7788 19-APR-87 01-MAY-87 01-APR-87
      7876 23-MAY-87 01-JUN-87 01-MAY-87
```

# Fonctions de Conversion




## Fonctions de Conversion

Il est possible d'utiliser des types de données ANSI, DB2 et SQL/DS, en plus des types de données Oracle, pour définir les colonnes d'une table dans une base de données Oracle8. Toutefois, Oracle8 Server convertit en interne ces types de données en types de données Oracle8.

Dans certains cas, Oracle8 peut accepter des données d'un type différent de celui normalement attendu, sous réserve qu'Oracle Server puisse effectuer une conversion automatique de ces données. Cette conversion des types de données est réalisée, soit de manière *implicite* par Oracle8 Server, soit de manière *explicite* par l'utilisateur.

Les conversions implicites de types de données fonctionnent selon des règles que nous allons expliquer dans les deux diapositives suivantes.

Les conversions explicites de types de données sont réalisées au moyen des fonctions de conversion. Ces fonctions convertissent une valeur d'un type de données en un autre type. En principe, le format de la fonction suit la convention *datatype* TO *datatype*, le premier *datatype* étant le type de données d'entrée, le second *datatype* étant le type de données restitué.

**Remarque :** Bien que la conversion implicite des types de données soit possible, il est recommandé d'effectuer des conversions explicites afin d'assurer une meilleure efficacité des ordres SQL.

# Conversion de Types de Données Implicite

Pour les affectations, Oracle effectue automatiquement les conversions suivantes

| De | Vers |
|---|---|
| VARCHAR2 ou CHAR | NUMBER |
| VARCHAR2 ou CHAR | DATE |
| NUMBER | VARCHAR2 |
| DATE | VARCHAR2 |

3-33

## Conversion de Types de Données Implicite

Pour les affectations, Oracle8 Server peut convertir automatiquement les types de données suivants :

- VARCHAR2 ou CHAR vers NUMBER
- VARCHAR2 ou CHAR vers DATE
- NUMBER vers VARCHAR2
- DATE vers VARCHAR2

L'affectation réussit si Oracle Server parvient à convertir le type de données de la valeur à affecter dans le type de données de la cible de l'affectation.

# Conversion de Types de Données Implicite

## Pour l'evaluation d'expressions, Oracle effectue automatiquement les conversions suivantes

| De | Vers |
|---|---|
| VARCHAR2 ou CHAR | NUMBER |
| VARCHAR2 ou CHAR | DATE |

3-23

### Conversion de Types de Données Implicite

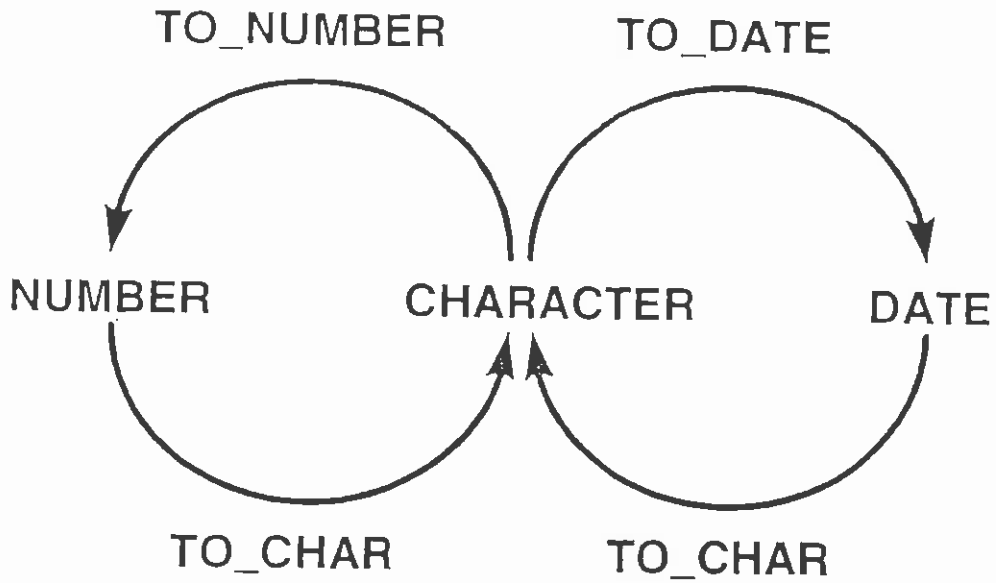
Pour l'évaluation d'expressions, Oracle Server peut convertir automatiquement les valeurs suivantes :

- VARCHAR2 ou CHAR vers NUMBER
- VARCHAR2 ou CHAR vers DATE

Pour des raisons de lisibilité, de performance et d'évolution ultérieure des applications Oracle, il est recommandé d'utiliser la conversion explicite des types de données.

Remarque : les conversions CHAR vers NUMBER ne sont possibles que lorsque le nombre représenté par la chaîne de caractères est valide. De même, les conversions VARCHAR2 vers DATE ne fonctionnent que si la chaîne de caractères est dans le format par défaut DD-MON-YY.

# Conversion de Types de Données Explicite




3-25

## Conversion de Types de Données Explicite

SQL offre trois fonctions pour convertir une valeur d'un certain type de données dans autre type.

| Fonction | Résultat |
|---|---|
| TO_CHAR(*number/date*,[*'fmt'*]) | Convertit un nombre ou une date en une chaîne de caractères de type VARCHAR2 et de format *fmt*. |
| TO_NUMBER(*char*) | Convertit une chaîne de caractères en un nombre. |
| TO_DATE(*char*,[*'fmt'*]) | Convertit une chaîne de caractères représentant une date au format *fmt* en une date Oracle. Lorsque *fmt* est omis, le format est DD-MON-YY. |

Remarque : cette liste de fonctions de conversion n'est pas exhaustive.

Pour plus d'informations, reportez-vous à *Oracle8 Server SQL Reference, Release 8*, "Conversion Functions."

# Utilisation de la Fonction TO_CHAR avec les Dates

```
TO_CHAR(date, 'fmt')
```

Le modèle de format :

- Doit être placé entre simples quotes et différencie les majuscules et minuscules.
- Peut inclure tout élément valide de format date
- Comporte un élément *fm* qui supprime les espaces de remplissage ou les zéros de tête
- Est séparé de la valeur date par une virgule

### Affichage d'une Date dans un Format Spécifique

Nous avons vu que Oracle Server affiche les dates au format DD-MON-YY. La fonction TO-CHAR permet de convertir ces dates en un autre format de votre choix.

### Conseils

- Le modèle de format doit être placé entre simples quotes et différencie les majuscules et minuscules.
- Il peut comprendre tout élément valide de format date. N'oubliez pas de séparer la date et le modèle de format par une virgule.
- Les noms de jours et de mois sont remplis par des espaces.
- Pour supprimer les espaces de remplissage ou [illegible] utilisez l'élément *fm*. *fm* signifie fill mode, ou mode de remplissage
- La colonne résultante a une largeur par défaut de 80 caractères.
- Vous pouvez redimensionner la largeur de la colonne résultante avec la commande SQL*Plus COLUMN.

```
SQL> SELECT   empno, TO_CHAR(hiredate, 'MM/YY') Month_Hired
  2  FROM     emp
  3  WHERE    ename = 'BLAKE';
```

# 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 |

3-27

Eléments de Format de Date Valides

| Elément | Description |
|---|---|
| SCC ou CC | Siècle; le S fait précéder les dates av. J.C. d'un signe - |
| YYYY ou SYYYY | Année; le S fait précéder les dates av. J.C. d'un signe - |
| YYY ou YY ou Y | Les 3, 2 ou 1 derniers chiffres de l'année |
| Y,YYY | Année avec une virgule insérée |
| IYYY, IYY, IY, I | Les 4, 3, 2 ou 1 derniers chiffres de l'année (norme ISO) |
| SYEAR ou YEAR | Année en toutes lettres ; le S fait précéder les dates av. J.C. d'un signe - |
| BC ou AD | Respectivement, av. JC ou apr. JC |
| B.C. ou A.D. | Respectivement, av. J.C. ou apr. J.C. |
| Q | Numéro du trimestre |
| MM | Mois exprimé avec 2 chiffres |
| MONTH | Mois en toutes lettres complété par des blancs à concurrence de 9 caractères |
| MON | 3 premières lettres du nom du mois |
| RM | Numéro du mois en chiffres romains |
| WW ou W | Numéro de la semaine dans l'année ou le mois |
| DDD ou DD ou D | Numéros du jour dans l'année, le mois ou la semaine |
| DAY | Nom du jour exprimé en toutes lettres et complété par des blancs à concurrence de 9 caractères |
| DY | 3 premières lettres du nom du jour |
| J | Jour du calendrier Julien ; nombre de jours depuis le 31 décembre 4713 av J.C. |

# Modèles de Format pour les Dates

- Les éléments horaires formatent la partie horaire de la date.

| | |
|---|---|
| HH24:MI:SS AM | 15:45:32 PM |

- Pour ajouter des chaînes de caractères, les placer entre guillemets.

| | |
|---|---|
| DD "of" MONTH | 12 of OCTOBER |

- Différents suffixes existent pour les nombres.

| | |
|---|---|
| ddspth | fourteenth |

3-27

### Formats Horaires

Utilisez les formats suivants pour afficher des informations et littéraux de type heure, et pour transformer des valeurs numériques en caractères.

| Elément | Description |
|---|---|
| AM ou PM | Respectivement, matin ou après-midi |
| A.M. ou P.M. | Respectivement, matin ou après-midi avec points |
| HH ou HH12 ou HH24 | [illegible] du jou[illegible] à 12) ou heure (0 à [illegible] |
| MI | Minutes (0 à 59) |
| SS | Secondes (0 à 59) |
| SSSSS | Secondes après [illegible] (0 à 86399) |

### Autres Formats

| Elément | Description |
|---|---|
| / , . ; | La ponctuation est reproduite dans le résultat |
| " of the " | Les chaînes entre guillemets sont reproduites telles quelles dans le résultat |

### Ajout de Suffixes pour Modifier L'Affichage des Données Numériques

| Elément | Description |
|---|---|
| TH | Nombre ordinal (par exemple, DDTH pour 4TH) |
| SP | Nombre en toutes lettres (par exemple, DDSP pour FOUR) |
| SPTH ou THSP | Nombres ordinaux en toutes lettres (par exemple, DDSPTH pour FOURTH) |

# Format de Date RR

| Année en Cours | Date Spécifiée | Format RR | Format YY |
|---|---|---|---|
| 1995 | 27-OCT-95 | 1995 | 1995 |
| 1995 | 27-OCT-17 | 2017 | 1917 |
| 2001 | 27-OCT-17 | 2017 | 2017 |
| 2001 | 27-OCT-95 | 1995 | 2095 |

| | | Si l'année spécifiée est située entre | |
|---|---|---|---|
| | | 0-49 | 50-99 |
| Si les 2 chiffres de l'année en cours sont | 0-49 | La nouvelle date appartient au siècle courant. | La nouvelle date appartient au siècle précédent. |
| | 50-99 | La nouvelle date appartient au siècle suivant. | La nouvelle date appartient au siècle courant. |

3-29

## Elément de Format de Date RR

Le format de date RR est identique à l'élément YY, mais permet en outre de changer de siècle. Vous pouvez l'utiliser à la place du format YY pour faire varier le siècle en fonction des 2 chiffres de l'année spécifiée et des deux derniers chiffres de l'année en cours. Le tableau de la diapositive résume le fonctionnement de l'élément RR.

| Année en cours | Date spécifiée | Interprétée RR | Interprétée YY |
|---|---|---|---|
| 1994 | 27-OCT-95 | 1995 | 1995 |
| 1994 | 27-OCT-17 | 2017 | 1917 |
| 2001 | 27-OCT-17 | 2017 | 2017 |

# 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.
```

3-30

### Utilisation de la Fonction TO_CHAR avec les Dates

L'ordre SQL ci-dessus affiche le nom et la date d'embauche de tous les employés. La date est affichée sous la forme 17 November 1981.

### Exemple

Modifier l'exemple ci-dessus afin d'obtenir l'affichage de la date dans le format suivant : Seventh of February 1981 08:00:00 AM.

```
SQL>SELECT ename,
  2         TO_CHAR(hiredate, 'fmDdspth "of" Month YYYY fmHH:MI:SS AM')
  3          HIREDATE
  4  FROM    emp;
```

```
ENAME      HIREDATE
---------- ------------------------------
KING       Seventeenth of November 1981 12:00:00 AM
BLAKE      First of May 1981 12:00:00 AM
...
14 rows selected.
```

Remarquez que le mois suit le modèle de format spécifié (INITCAP).

# TO_CHAR avec les Nombres

```
TO_CHAR(number, 'fmt')
```

Utilisez les formats suivants avec TO_CHAR pour afficher un nombre sous la forme d'une chaîne de caractère.

| | |
|---|---|
| 9 | Représente un chiffre |
| 0 | Force l'affichage du zéro |
| $ | Place un signe dollar flottant |
| L | Utilise le symbole monétaire local flottant |
| . | Imprime un point décimal |
| , | Imprime un séparateur de milliers |

3-31

## Utilisation de la Fonction TO_CHAR avec les Nombres

Pour pouvoir afficher des valeurs numériques sous la forme de chaînes de caractères, il convient de convertir ces nombres en données de type caractère avec la fonction TO_CHAR, qui transforme une valeur de type NUMBER en un type VARCHAR2. Cette technique est très utile pour la concaténation.

## Eléments de Format Numérique

Pour convertir un nombre en un type caractère, utilisez les éléments suivants :

| Elément | Description | Exemple | Résultat |
|---|---|---|---|
| 9 | Le nombre de 9 détermine la largeur maximum de l'affichage | 999999 | 1234 |
| 0 | Affichage des zéros de gauche | 099999 | 001234 |
| $ | Signe dollar flottant | $99999 | $1234 |
| L | Symbole monétaire local flottant | L99999 | FF1234 |
| . | Point décimal à l'emplacement spécifié | 99999.99 | 1234.00 |
| , | Séparateur de milliers à l'emplacement spécifié | 999,999 | 1,234 |
| MI | Signe moins à droite (valeurs négatives) | 999999MI | 1234- |
| PR | Place les nombres négatifs entre crochets angulaires | 999999PR | <1234> |
| EEEE | Notation scientifique (indiquer obligatoirement quatre E) | 99.999EEEE | 1.234E-03 |
| V | Multiplie par 10 n fois (n = nombre de chiffres après V) | 9999V99 | 123400 |
| B | N'affiche pas les zéros non significatifs | B9999.99 | 1234.00 |

# Utilisation de la Fonction TO_CHAR avec les Nombres

```
SQL> SELECT   TO_CHAR(sal,'$99,999') SALARY
  2  FROM     emp
  3  WHERE    ename = 'SCOTT';
```

```
SALARY
--------
 $3,000
```

### Conseils

- Oracle Server affiche une chaîne de signes dièse (#) lorsque le nombre de chiffres de la valeur excède le nombre de chiffres spécifié dans le modèle de format.
- Oracle Server arrondit la valeur décimale au nombre de décimales spécifié dans le modèle de format.

- Conversion d'une chaîne de caractères en format numérique avec la fonction TO_NUMBER

```
TO_NUMBER(char)
```

- Conversion d'une chaîne de caractères en format date avec la fonction TO_DATE

```
TO_DATE(char[, 'fmt'])
```

3-33

## Fonctions TO_NUMBER et TO_DATE

Vous pouvez convertir une chaîne de caractères en format numérique ou date en utilisant respectivement les fonctions TO_NUMBER ou TO_DATE. Pour TO_DATE, le modèle de format à spécifier est basé sur les éléments de format déjà expliqués. Ce modèle de format doit décrire le format de la chaîne fournie en entrée.

### Exemple

Afficher le nom et la date d'embauche de tous les employés entrés le "February 22, 1981".

```
SQL> SELECT ename, hiredate
  2  FROM   emp
  3  WHERE  hiredate = TO_DATE('February 22, 1981', 'Month dd, YYYY');
```

```
ENAME      HIREDATE
---------- --------
WARD       22-FEB-81
```

# 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')

3-34

### La fonction NVL

Pour transformer une valeur NULL en une valeur réelle, on utilise la fonction NVL.

### Syntaxe

```
NVL (expr1, expr2)
```

où : *expr1* est l'expression ou la valeur source suceptible de contenir une valeur NULL

*expr2* est la valeur de remplacement pour la valeur NULL

La fonction NVL permet de convertir n'importe quel type de données, mais toutefois, la valeur de remplacement doit être de même type que la valeur de l'expression *expr1*.

### Conversions NVL pour Divers Types de Données

| Type de données | Exemple de conversion |
|---|---|
| NUMBER | NVL(*number_column*,9) |
| DATE | NVL(*date_column*, '01-JAN-95') |
| CHAR ou VARCHAR2 | NVL(*character_column*, 'Unavailable') |

# 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.
```



## Fonction NVL

Pour calculer la rémunération annuelle de chaque employé, il faut multiplier son salaire mensuel par 12 puis ajouter la commission au résultat obtenu.

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

```
ENAME      JOB       (SAL*12)+COMM
---------- --------- -------------
KING       PRESIDENT
BLAKE      MANAGER
CLARK      MANAGER
JONES      MANAGER
MARTIN     SALESMAN          16400
...
14 rows selected.
```

Notez que le résultat de ce calcul de rémunération annuelle est renseigné uniquement pour les employés qui perçoivent une commission. Lorsqu'un argument dans une expression arithmétique est NULL, le résultat de cette expression sera NULL. Pour calculer la rémunération annuelle de tous les employés, il faut convertir la valeur NULL en une valeur numérique avant d'appliquer l'opérateur arithmétique. Dans l'exemple ci-dessus, la fonction NVL est utilisée pour convertir les valeurs NULL en zéro.

# Fonction DECODE

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

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

### La Fonction DECODE

La fonction DECODE permet de décoder les expressions de la même manière que l'ordre logique IF-THEN-ELSE utilisé dans de nombreux langages. Elle décode l'*expression* après l'avoir comparée à chacune des valeurs de recherche (*search*). Si l'expression est identique à *search*, le résultat (*result*) est ramené.

Si la valeur par défaut (*default*) est omise, on obtient une valeur NULL chaque fois que la colonne ou expression ne correspond à aucune valeur *search*.

# Utilisation de la Fonction DECODE

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

```
JOB              SAL REVISED_SALARY
--------- --------- --------------
PRESIDENT       5000           5000
MANAGER         2850           3420
MANAGER         2450           2940
...
14 rows selected.
```

3-37

## Utilisation de la Fonction DECODE

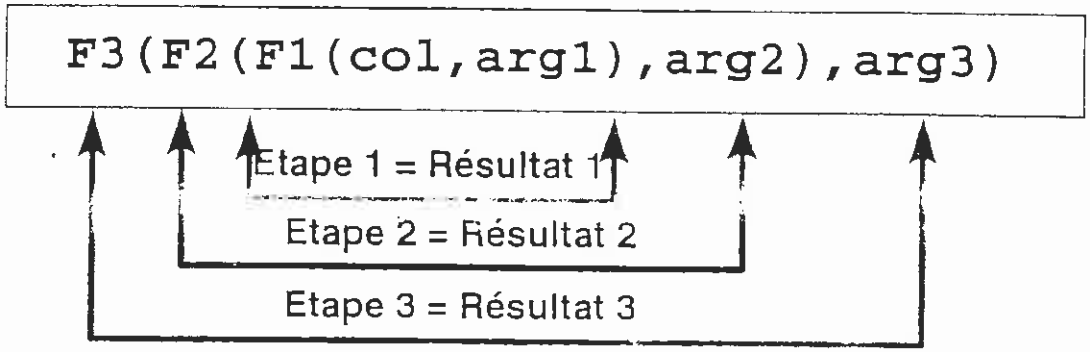
Dans l'ordre SQL ci-dessus, la valeur de JOB est décodée. Si la valeur de JOB correspond à ANALYST, alors l'augmentation de salaire est de 10% ; si elle correspond à CLERK, elle est de 15% et si elle correspond à MANAGER, elle est de 20%. Il n'y a aucune augmentation de salaire pour toutes les autres valeurs de JOB.

Cette instruction correspondrait à l'ordre IF-THEN-ELSE suivant :

```
IF job = 'ANALYST'    THEN  sal = sal*1.1
IF job = 'CLERK'      THEN  sal = sal*1.15
IF job = 'MANAGER'    THEN  sal = sal*1.20
ELSE sal = sal
```

# Imbrication des Fonctions

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




### Imbrication des Fonctions

Le niveau d'imbrication des fonctions mono-ligne est illimité. L'évaluation se fait du niveau le plus interne vers le niveau le plus externe. Les exemples qui suivent montrent la flexibilité de ces fonctions.

# 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
```

3-39

## Imbrication des Fonctions

L'exemple ci-dessus affiche le nom du directeur de l'entreprise (qui n'a pas de manager).
L'évaluation de l'ordre SQL se fait en deux temps :

1. Evaluation de la fonction interne qui convertit une valeur numérique en chaîne de caractères.
   - Result1 = TO_CHAR(mgr)
2. Evaluation de la fonction externe qui remplace la valeur NULL par une chaîne de texte.
   - NVL(Result1, 'No Manager')

L'expression entière devient l'en-tête de colonne puisqu'aucun alias de colonne n'a été donné.

## Exemple

Afficher la date correspondant au premier vendredi qui tombe six mois après la date d'embauche. La date résultante doit se présenter comme suit : Friday, March 12th, 1982. Classer les résultats par date d'embauche.

```
SQL> SELECT    TO_CHAR(NEXT_DAY(ADD_MONTHS
  2            (hiredate, 6), 'FRIDAY'),
  3            'fmDay, Month ddth, YYYY')
  4            "Next 6 Month Review"
  5  FROM      emp
  6  ORDER BY  hiredate;
```

# Résumé

Utilisez des fonctions mono-ligne pour :

- Transformer des données
- Formater des dates et des nombres pour l'affichage
- Convertir des types de données de colonnes

3-40

### Fonctions Mono-Ligne

Les fonctions mono-ligne peuvent être imbriquées à l'infini. Elles peuvent manipuler :

- Des données caractères
  - LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- Des données numériques
  - ROUND, TRUNC, MOD
- Des dates
  - MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC
  - Il est aussi possible d'utiliser les opérateurs arithmétiques + et - sur des données de type DATE.
- Les fonctions de conversion agissent sur les données de type caractère, date et numérique.
  - TO_CHAR, TO_DATE, TO_NUMBER

### SYSDATE et DUAL

SYSDATE est une fonction date qui ramène la date et l'heure courantes. SYSDATE est généralement sélectionnée dans une table factice appelée DUAL.

# Présentation des Exercices

- Création de requêtes utilisant les fonctions numériques, caractère et date
- Utilisation de la concaténation avec les fonctions
- Ecriture de requêtes insensibles à la casse pour illustrer l'utilité des fonctions de type caractère
- Calcul des années et mois d'ancienneté d'un employé
- Détermination de la date de révision de salaire d'un employé

3-41

## Présentation des Exercices

La variété des exercices qui suivent est destinée à vous permettre de mettre en pratique les différentes fonctions utilisables avec des données de type caractère, numérique et date.

Rappelez-vous que dans les fonctions imbriquées, l'évaluation se fait de la fonction la plus interne vers la fonction la plus externe.

## Exercices 3

1. Ecrivez une requête pour afficher la date courante. Nommez la colonne Date.

```
Date
---------
28-OCT-97
```

2. Pour chaque employé, affichez le matricule, le nom, le salaire et le salaire augmenté de 15% sous la forme d'un nombre entier. Nommez cette colonne New Salary. Enregistrez votre ordre SQL dans un fichier appelé *p3q2.sql*.

3. Exécutez votre requête à partir du fichier *p3q2.sql*.

```
EMPNO ENAME      SAL New Salary
----- ------- ----- ----------
 7839 KING     5000       5750
 7698 BLAKE    2850       3278
 7782 CLARK    2450       2818
 7566 JONES    2975       3421
 7654 MARTIN   1250       1438
 7499 ALLEN    1600       1840
 7844 TURNER   1500       1725
 7900 JAMES     950       1093
 7521 WARD     1250       1438
 7902 FORD     3000       3450
 7369 SMITH     800        920
 7788 SCOTT    3000       3450
 7876 ADAMS    1100       1265
 7934 MILLER   1300       1495
14 rows selected.
```

4. Modifiez votre requête *p3q2.sql* en ajoutant une colonne dans laquelle l'ancien salaire est soustrait du nouveau salaire. Nommez cette colonne Increase. Exécutez à nouveau votre requête.

```
EMPNO ENAME      SAL New Salary Increase
----- ------ ----- ---------- --------
 7839 KING     5000       5750      750
 7698 BLAKE    2850       3278      428
 7782 CLARK    2450       2818      368
 7566 JONES    2975       3421      446
```

5 Affichez le nom et la date d'embauche de chaque employé ainsi que la date de révision du salaire qui sera le premier lundi tombant après 6 mois d'activité. Nommez la colonne REVIEW. Les dates devront apparaître dans le format suivant : "Sunday, the Seventh of September, 1981."

```
ENAME   HIREDATE  REVIEW
-----   --------  ----------------------------------------
KING    17-NOV-81 Monday, the Twenty-Fourth of May, 1982
BLAKE   01-MAY-81 Monday, the Second of November, 1981
CLARK   09-JUN-81 Monday, the Fourteenth of December, 1981
JONES   02-APR-81 Monday, the Fifth of October, 1981
MARTIN  28-SEP-81 Monday, the Twenty-Ninth of March, 1982
ALLEN   20-FEB-81 Monday, the Twenty-Fourth of August, 1981
TURNER  08-SEP-81 Monday, the Fifteenth of March, 1982
JAMES   03-DEC-81 Monday, the Seventh of June, 1982
WARD    22-FEB-81 Monday, the Twenty-Fourth of August, 1981
FORD    03-DEC-81 Monday, the Seventh of June, 1982
SMITH   17-DEC-80 Monday, the Twenty-Second of June, 1981
SCOTT   09-DEC-82 Monday, the Thirteenth of June, 1983
ADAMS   12-JAN-83 Monday, the Eighteenth of July, 1983
MILLER  23-JAN-82 Monday, the Twenty-Sixth of July, 1982

14 rows selected.
```

6. Affichez le nom de chaque employé et calculez le nombre de mois travaillés depuis la date d'embauche. Nommez la colonne MONTHS_WORKED. Classez les résultats en fonction du nombre de mois d'ancienneté. Arrondissez le nombre de mois à l'entier le plus proche.

```
ENAME   MONTHS_WORKED
------- -------------
ADAMS           177
SCOTT           178
MILLER          188
JAMES           190
FORD            190
KING            191
MARTIN          192
TURNER          193
CLARK           196
BLAKE           197
JONES           198
WARD            199
ALLEN           199
SMITH           202

14 rows selected.
```

7. Ecrivez une requête affichant les informations suivantes pour chaque employé : <employee name> earns <salary> monthly but wants <3 times salary>. Nommez la colonne Dream Salaries.

```
Dream Salaries
--------------------------------------------------
KING earns $5,000.00 monthly but wants $15,000.00.
BLAKE earns $2,850.00 monthly but wants $8,550.00.
CLARK earns $2,450.00 monthly but wants $7,350.00.
JONES earns $2,975.00 monthly but wants $8,925.00.
MARTIN earns $1,250.00 monthly but wants $3,750.00.
ALLEN earns $1,600.00 monthly but wants $4,800.00
TURNER earns $1,500.00 monthly but wants $4,500.00.
JAMES earns $950.00 monthly but wants $2,850.00.
WARD earns $1,250.00 monthly but wants $3,750.00.
FORD earns $3,000.00 monthly but wants $9,000.00.
SMITH earns $800.00 monthly but wants $2,400.00.
SCOTT earns $3,000.00 monthly but wants $9,000.00.
ADAMS earns $1,100.00 monthly but wants $3,300.00
MILLER earns $1,300.00 monthly but wants $3,900.00.
14 rows selected.
```

Si vous avez le temps, faites les exercices suivants :

8. Créez une requête pour afficher le nom et le salaire de tous les employés. Le salaire sera formaté de façon à avoir 15 caractères de long, la valeur du salaire étant complétée à gauche par des $. Nommez la colonne SALARY.

```
ENAME      SALARY
---------- ---------------
SMITH      $$$$$$$$$$$800
ALLEN      $$$$$$$$$$1600
WARD       $$$$$$$$$$1250
JONES      $$$$$$$$$$2975
MARTIN     $$$$$$$$$$1250
BLAKE      $$$$$$$$$$2850
CLARK      $$$$$$$$$$2450
SCOTT      $$$$$$$$$$3000
KING       $$$$$$$$$$5000
TURNER     $$$$$$$$$$1500
ADAMS      $$$$$$$$$$1100
JAMES      $$$$$$$$$$950
FORD       $$$$$$$$$$3000
MILLER     $$$$$$$$$$1300
14 rows selected.
```

9. Ecrivez une requête pour afficher tous les noms d'employé commençant par les lettres *J*, *A*, ou *M*, ainsi que la longueur du nom. Le nom doit apparaître en minuscules, sauf l'initiale qui sera en majuscules. Donnez à chaque colonne un nom approprié.

```
Name     Length
------- ------
Jones         5
Martin        6
Allen         5
James         5
Adams         5
Miller        6
6 rows selected.
```

10. Affichez le nom, la date d'embauche ainsi que le jour de la semaine où l'employé à débuté. Nommez la colonne DAY. Classez les résultats dans l'ordre des jours de la semaine à partir du lundi (monday).

```
ENAME    HIREDATE  DAY
-------  --------- ----------
MARTIN   28-SEP-81 MONDAY
CLARK    09-JUN-81 TUESDAY
KING     17-NOV-81 TUESDAY
TURNER   08-SEP-81 TUESDAY
SMITH    17-DEC-80 WEDNESDAY
ADAMS    12-JAN-83 WEDNESDAY
JONES    02-APR-81 THURSDAY
FORD     03-DEC-81 THURSDAY
SCOTT    09-DEC-82 THURSDAY
JAMES    03-DEC-81 THURSDAY
ALLEN    20-FEB-81 FRIDAY
BLAKE    01-MAY-81 FRIDAY
MILLER    23-JAN-82 SATURDAY
WARD     22-FEB-81 SUNDAY
14 rows selected
```

## Exercice 3

Si vous souhaitez aller plus loin dans la difficulté, faites les exercices suivants :

11. Créez une requête pour afficher le nom et le montant de la commission de chaque employé. Pour les employés ne touchant aucune commission, affichez "No Commission". Nommez la colonne COMM.

```
ENAME      COMM
------     -----------
SMITH      No Commission
ALLEN      300
WARD       500
JONES      No Commission
MARTIN     1400
BLAKE      No Commission
CLARK      No Commission
SCOTT      No Commission
KING       No Commission
TURNER     0
ADAMS      No Commission
JAMES      No Commission
FORD       No Commission
MILLER     No Commission
14 rows selected.
```

1. Ecrivez une requête pour afficher le nom, le numéro de département et le département de tous les employés.

```
ENAME      DEPTNO DNAME
------ ---------- ----------
CLARK          10 ACCOUNTING
KING           10 ACCOUNTING
MILLER         10 ACCOUNTING
SMITH          20 RESEARCH
ADAMS          20 RESEARCH
FORD           20 RESEARCH
SCOTT          20 RESEARCH
JONES          20 RESEARCH
ALLEN          30 SALES
BLAKE          30 SALES
MARTIN         30 SALES
JAMES          30 SALES
TURNER         30 SALES
WARD           30 SALES
14 rows selected.
```

2. Créez une liste unique de tous les postes du département 30.

```
JOB       LOC
--------- -------------
CLERK     CHICAGO
MANAGER   CHICAGO
SALESMAN  CHICAGO
```

3. Ecrivez une requête pour afficher le nom, le nom du département et la localisation de tous les employés qui touchent une commission.

```
ENAME  DNAME  LOC
------ ------ -------------
ALLEN  SALES  CHICAGO
WARD   SALES  CHICAGO
MARTIN SALES  CHICAGO
TURNER SALES  CHICAGO
```

## Exercices 4

4. Affichez le nom et le nom du département pour tous les employés dont le nom contient la lettre A. Enregistrez votre ordre SQL dans un fichier que vous nommerez *p4q4.sql*.

```
ENAME      DNAME
-------    ----------
CLARK      ACCOUNTING
ADAMS      RESEARCH
ALLEN      SALES
WARD       SALES
JAMES      SALES
MARTIN     SALES
BLAKE      SALES
7 rows selected.
```

5. Ecrivez une requête pour afficher le nom, le poste, le numéro de département et le nom du département de tous les employés basés à DALLAS.

```
ENAME    JOB       DEPTNO DNAME
-------  --------- ------ ----------
SMITH    CLERK         20 RESEARCH
ADAMS    CLERK         20 RESEARCH
FORD     ANALYST       20 RESEARCH
SCOTT    ANALYST       20 RESEARCH
JONES    MANAGER       20 RESEARCH
```

6. Affichez le nom et le matricule des employés et de leur manager. Nommez les colonnes Employee, Emp#, Manager, et Mgr#, respectivement. Enregistrez votre ordre SQL dans un fichier nommé *p4q6.sql*.

```
Employee     Emp# Manager Mgr#
--------   ------ ------- -----
SCOTT        7788 JONES    7566
FORD         7902 JONES    7566
ALLEN        7499 BLAKE    7698
WARD         7521 BLAKE    7698
JAMES        7900 BLAKE    7698
TURNER       7844 BLAKE    7698
MARTIN       7654 BLAKE    7698
MILLER       7934 CLARK    7782
ADAMS        7876 SCOTT    7788
JONES        7566 KING     7839
CLARK        7782 KING     7839
BLAKE        7698 KING     7839
SMITH        7369 FORD     7902
13 rows selected.
```

7. Modifiez le fichier *p4q6.sql* pour afficher tous les employés, y compris King, n'ayant pas de manager. Enregistrez à nouveau dans un fichier *p4q7.sql*. Exécutez *p4q7.sql*.

```
Employee   Emp# Manager  Mgr#
-------- ------ ------- -----
SCOTT      7788 JONES     7566
FORD       7902 JONES     7566
ALLEN      7499 BLAKE     7698
WARD       7521 BLAKE     7698
JAMES      7900 BLAKE     7698
TURNER     7844 BLAKE     7698
MARTIN     7654 BLAKE     7698
MILLER     7934 CLARK     7782
ADAMS      7876 SCOTT     7788
JONES      7566 KING      7839
CLARK      7782 KING      7839
BLAKE      7698 KING      7839
SMITH      7369 FORD      7902
KING       7839
14 rows selected.
```

Si vous avez le temps, faites les exercices suivants :

8. Créez une requête pour afficher le numéro de département et le nom de tous les employés qui travaillent dans le même département qu'un certain employé. Donnez à chaque colonne un en-tête approprié.

```
DEPARTMENT EMPLOYEE COLLEAGUE
---------- -------- ---------
        10 CLARK    KING
        10 CLARK    MILLER
        10 KING     CLARK
        10 KING     MILLER
        10 MILLER   CLARK
        10 MILLER   KING
        20 ADAMS    FORD
        20 ADAMS    JONES
        20 ADAMS    SCOTT
        20 ADAMS    SMITH
        20 FORD     ADAMS
        20 FORD     JONES
        20 FORD     SCOTT
...
56 rows selected.
```



3-9

9. Affichez la structure de la table SALGRADE. Créez une requête pour afficher le nom, le poste, le département, le salaire et l'échelon de tous les employés.

```
Name          NULL?    Type
------------- -------- ----
GRADE                  NUMBER
LOSAL                  NUMBER
HISAL                  NUMBER
```

```
ENAME   JOB         DNAME          SAL GRADE
------  ----------  ---------- ----- -----
MILLER  CLERK       ACCOUNTING  1300     2
CLARK   MANAGER     ACCOUNTING  2450     4
KING    PRESIDENT   ACCOUNTING  5000     5
SMITH   CLERK       RESEARCH     800     1
SCOTT   ANALYST     RESEARCH    3000     4
FORD    ANALYST     RESEARCH    3000     4
ADAMS   CLERK       RESEARCH    1100     1
JONES   MANAGER     RESEARCH    2975     4
JAMES   CLERK       SALES        950     1
BLAKE   MANAGER     SALES       2850     4
TURNER  SALESMAN    SALES       1500     3
ALLEN   SALESMAN    SALES       1600     3
WARD    SALESMAN    SALES       1250     2
MARTIN  SALESMAN    SALES       1250     2
14 rows selected.
```

Si vous souhaitez aller plus loin dans la difficulté, faites les exercices suivants :

10. Créez une requête pour afficher le nom et la date d'embauche de tous les employés arrivés après l'employé Blake.

```
ENAME   HIREDATE
------- ---------
SMITH   17-DEC-80
ALLEN   20-FEB-81
WARD    22-FEB-81
JONES   02-APR-81
```

11. Affichez les noms et date d'embauche des employés et de leur manager, pour tous les employés ayant été embauchés avant leur manager. Nommez les colonnes Employee, Emp Hiredate, Manager et Mgr Hiredate, respectivement.

```
Employee Emp Hiredate  Manager Mgr Hiredate
-------- ------------- ------- ------------
ALLEN    20-FEB-81     BLAKE   01-MAY-81
WARD     22-FEB-81     BLAKE   01-MAY-81
JONES    02-APR-81     KING    17-NOV-81
CLARK    09-JUN-81     KING    17-NOV-81
BLAKE    01-MAY-81     KING    17-NOV-81
SMITH    17-DEC-80     FORD    03-DEC-81
6 rows selected.
```

12. Créez une requête pour afficher le nom des employés et leur salaire indiqué par des astérisques. Chaque astérisque représente cent dollars. Triez les données dans l'ordre décroissant des salaires. Nommez la colonne EMPLOYEE_AND_THEIR_SALARIES.

```
EMPLOYEE_AND_THEIR_SALARIES
---------------------------------------------------------------
KING      ***********************************************
FORD      ****************************
SCOTT     ****************************
JONES     ****************************
BLAKE     ************************
CLARK     ************************
ALLEN     **************
TURNER    **************
MILLER    ************
MARTIN    ************
WARD      ************
ADAMS     **********
JAMES     ********
SMITH     *******
14 rows selected.
```

## Exercices 5

Déterminez si les affirmations suivantes sont vraies ou fausses et entourez la réponse correspondante.

1. Les fonctions de groupe agissent sur plusieurs lignes pour produire un seul résultat.
   Vrai/Faux
2. Les fonctions de groupe intègrent les valeurs NULL dans leurs calculs.
   Vrai/Faux
3. La clause WHERE restreint les lignes avant qu'elles soient incluses dans un calcul de groupe.
   Vrai/Faux

4. Affichez le salaire maximum, le salaire minimum, la somme des salaires et le salaire moyen de tous les employés. Nommez respectivement les colonnes Maximum, Minimum, Sum et Average. Arrondissez les résultats à zéro décimale. Enregistrez votre ordre SQL dans un fichier nommé *p5q4.sql*.

```
 Maximum   Minimum     Sum Average
--------  -------- ------ -------
    5000       800  29025    2073
```

5. Modifiez le fichier *p5q4.sql* pour afficher le salaire maximum, le salaire minimum, la somme des salaires et le salaire moyen pour chaque type de poste.
   Enregistrez votre fichier sous *p5q5.sql*. Exécutez à nouveau votre requête.

```
JOB         Maximum  Minimum      Sum  Average
---------- -------- -------- ------- --------
ANALYST        3000     3000    6000     3000
CLERK          1300      800    4150     1038
MANAGER        2975     2450    8275     2758
PRESIDENT      5000     5000    5000     5000
SALESMAN       1600     1250    5600     1400
```

6. Ecrivez une requête pour afficher le nombre de personnes qui occupent le même poste.

```
JOB         COUNT(*)
---------- --------
ANALYST            2
CLERK              4
MANAGER            3
PRESIDENT          1
SALESMAN           4
```



3-12

## Exercices 5

7. Déterminez le nombre de managers sans en donner la liste. Nommez la colonne Number of Managers.

```
Number of Managers
------------------
                 6
```

8. Ecrivez une requête pour afficher la différence existant entre le salaire maximum et le salaire minimum. Nommez la colonne DIFFERENCE.

```
DIFFERENCE
----------
      4200
```

Si vous avez le temps, faites les exercices suivants :

9. Affichez le matricule des différents managers et le niveau de salaire le plus bas de leurs employés.
Excluez toute ligne où le manager n'est pas identifié. Excluez tout groupe dans lequel le salaire minimum est inférieur à $1000. Triez les résultats par ordre décroissant des salaires.

```
       MGR   MIN(SAL)
---------- ----------
      7566       3000
      7839       2450
      7782       1300
      7788       1100
```

10. Ecrivez une requête pour afficher le nom du département, la localisation, le nombre d'employés et le salaire moyen pour tous les employés de ce département. Nommez les colonnes dname, loc, Number of People et Salary, respectivement.

```
DNAME          LOC       Number of People  Salary
-------------- --------- ---------------- -------
ACCOUNTING     NEW YORK                3 2916.67
RESEARCH       DALLAS                  5    2175
SALES          CHICAGO                 6 1566.67
```

## Exercices 5

Si vous souhaitez aller plus loin dans la difficulté, faites les exercices suivants :

11. Créez une requête pour afficher le nombre total d'employés puis, parmi ces employés, ceux qui ont été embauchés en 1980, 1981, 1982 et 1983. Nommez les colonnes de façon appropriée.

| TOTAL | 1980 | 1981 | 1982 | 1983 |
|---|---|---|---|---|
| 14 | 1 | 10 | 2 | 1 |

12. Créez une requête pour afficher les postes, le salaire de ces postes par numéro de département et le salaire total de ces postes incluant tous les départements. Nommez les colonnes de façon appropriée.

| Job | Dept 10 | Dept 20 | Dept 30 | Total |
|---|---|---|---|---|
| ANALYST | | 6000 | | 6000 |
| CLERK | 1300 | 1900 | 950 | 4150 |
| MANAGER | 2450 | 2975 | 2850 | 8275 |
| PRESIDENT | 5000 | | | 5000 |
| SALESMAN | | | 5600 | 5600 |

# Présentation des Exercices

Dans ces exercices, vous allez écrire des requêtes en incluant des opérateurs ensemblistes.

- Utilisation d'autres méthodes de jointure
- Ecriture de requêtes composées sous forme d'ordres IF

6-17

**Remarque :** Pour créer la table EMP_HISTORY, exécuter le script *emphis.sql*.

## Exercice 6

1. Affichez le département qui ne comprend aucun employé.

```
    DEPTNO DNAME
---------- --------------
        40 OPERATIONS
```

2. Retrouvez le poste qui était engagé dès la 1ère moitié des années 1981 et 1982.

```
JOB
---------
ANALYST
```

3. Ecrivez une requête composée pour produire une liste de produits indiquant les pourcentages de remise, les identifiants des produits, ainsi que les prix réels nouveaux et anciens. Les produits dont le prix est inférieur à $10 sont réduits de 10%, ceux dont le prix est compris entre $10 et $30 sont réduits de 15%, ceux dont le prix est supérieur à $30 sont réduits de 20%, et ceux dont le prix est supérieur à $40 ne sont pas réduits.

PAS LA TABLE

```
DISCOUNT     PRODID  STDPRICE  ACTPRICE
-------- --------- --------- ---------
10% off      100870       2.4      2.16
10% off      100870       2.8      2.52
10% off      100871       4.8      4.32
10% off      100871       5.6      5.04
10% off      102130       3.4      3.06
10% off      200376       2.4      2.16
10% off      200380         4       3.6
15% off      100860        30      25.5
15% off      101860        24      20.4
15% off      101863      12.5    10.625
20% off      100860        32      25.6
20% off      100860        35        28
20% off      100861        39      31.2
no disc      100861        42        42
no disc      100861        45        45
no disc      100890        54        54
no disc      100890        58        58
```

Exercice 6

4. Affichez la liste des postes dans les départements 10, 30 et 20, en conservant cet ordre. Affichez le poste et le numéro du département.

   Note : La commande SQL*Plus suivante permet de ne pas afficher la colonne DUMMY (DUMMY est le nom d'une colonne ramenée par le SELECT) : COL dummy NOPRINT PAS CONSEILLÉ

```
JOB           DEPTNO
--------- --------
CLERK             10
MANAGER           10
PRESIDENT         10
CLERK             30
MANAGER           30
SALESMAN          30
ANALYST           20
CLERK             20
MANAGER           20
```

5. Affichez le numéro des départements dans lesquels on ne trouve pas de poste ANALYST .

```
DEPTNO
------
    10
    30
    40
```

6. Affichez tous les postes des départements 10 et 20 qui n'existent que dans l'un ou l'autre de ces départements.

```
JOB
---------
ANALYST
PRESIDENT
```

## Exercices 7

1 Créez une requête pour afficher le nom et la date d'embauche de tous les employés travaillant dans le même département que Blake, à l'exclusion de Blake.

```
ENAME      HIREDATE
---------- ---------
MARTIN     28-SEP-81
ALLEN      20-FEB-81
TURNER     08-SEP-81
JAMES      03-DEC-81
WARD       22-FEB-81
6 rows selected.
```

2. Créez une requête pour afficher le matricule et le nom de tous les employés qui gagnent plus que le salaire moyen. Triez les résultats par ordre décroissant des salaires.

```
EMPNO ENAME
----- ----------
 7839 KING
 7902 FORD
 7788 SCOTT
 7566 JONES
 7698 BLAKE
 7782 CLARK
6 rows selected.
```

3 Ecrivez une requête pour afficher le matricule et le nom de tous les employés qui travaillent dans le même département que tout employé dont le nom contient un *T*. Enregistrez votre ordre SQL dans un fichier nommé *p6q3..sql*.

```
 EMPNO ENAME
------ --------
  7566 JONES
  7788 SCOTT
  7876 ADAMS
  7369 SMITH
       7902 FORD
  7698 BLAKE
  7654 MARTIN
  7499 ALLEN
  7844 TURNER
  7900 JAMES
  7521 WARD

11 rows selected.
```

4. Affichez le nom, le numéro de département et le poste de tous les employés dont le département est situé à Dallas.

```
ENAME   DEPTNO JOB
------ ------ ---------
JONES       20 MANAGER
FORD        20 ANALYST
SMITH       20 CLERK
SCOTT       20 ANALYST
ADAMS       20 CLERK
```

5. Affichez le nom et le salaire de tous les employés dont le manager est King.

```
ENAME       SAL
------- ------
BLAKE      2850
CLARK      2450
JONES      2975
```

6. Affichez le numéro de département, le nom et le poste de tous les employés travaillant dans le département des ventes ('SALES').

```
DEPTNO ENAME    JOB
------ -------- ---------
    30 BLAKE    MANAGER
    30 MARTIN   SALESMAN
    30 ALLEN    SALESMAN
    30 TURNER   SALESMAN
    30 JAMES    CLERK
    30 WARD     SALESMAN
6 rows selected.
```

Si vous avez le temps, faites les exercices suivants.

7. Modifiez *p6q3.sql* afin d'afficher le matricule, le nom et le salaire de tous les employés qui gagnent plus que le salaire moyen et qui travaillent dans un département avec tout employé dont le nom contient un *T*. Enregistrez à nouveau votre requête sous le nom *p6q7.sql*, puis réexécutez-la.

```
EMPNO ENAME      SAL
----- ------ ------
 7566 JONES    2975
 7788 SCOTT    3000
 7902 FORD     3000
 7698 BLAKE    2850
```

# Présentation des Exercices

## Création de sous-interrogations multi-colonne

8-13

### Présentation des Exercices

Au cours de cette série d'exercices, vous allez écrire des sous-interrogations multi-colonne.

## Exercices 8

1. Ecrivez une requête pour afficher le nom, le numéro de département et le salaire de tout employé dont le numéro de département et le salaire sont tous les deux à la fois équivalents au numéro de département et au salaire de n'importe quel employé touchant une commission.

```
ENAME      DEPTNO      SAL
---------- ------ ------
MARTIN         30   1250
WARD           30   1250
TURNER         30   1500
ALLEN          30   1600
```

2. Affichez le nom, le numéro de département et le salaire de tout employé dont le ~~numéro de département~~ et le salaire sont tous les deux à la fois équivalents au salaire et à la commission de n'importe quel employé basé à Dallas.

```
ENAME   DNAME          SAL
------- ---------- ------
SMITH   RESEARCH        800
ADAMS   RESEARCH       1100
JONES   RESEARCH       2975
FORD    RESEARCH       3000
SCOTT   RESEARCH       3000
```

3. Créez une requête pour afficher le nom, la date d'embauche et le salaire pour tous les employés touchant le même salaire et la même commission que Scott.

```
ENAME   HIREDATE      SAL
------- --------- ------

FORD    03-DEC-81    3000
```

4. Créez une requête pour afficher les employés qui perçoivent un salaire supérieur à tout employé dont le poste est CLERK. Triez le résultat par ordre décroissant des salaires.

```
ENAME      JOB              SAL
---------- --------- ---------
KING       PRESIDENT       5000
FORD       ANALYST         3000
SCOTT      ANALYST         3000
JONES      MANAGER         2975
BLAKE      MANAGER         2850
CLARK      MANAGER         2450
ALLEN      SALESMAN        1600
TURNER     SALESMAN        1500
8 rows selected.
```

## Exercice 9

1. Ecrivez une requête pour afficher les trois meilleurs salaires dans la table EMP. Affichez les noms des employés et leur salaire.

```
ENAME                SAL
---------- ---------
KING                5000
FORD                3000
SCOTT               3000
```

2. Recherchez tous les employés qui ne sont pas des responsables.

   a. Utilisez d'abord l'opérateur EXISTS.

```
ENAME
----------
MARTIN
ALLEN
TURNER
JAMES
WARD
SMITH
ADAMS
MILLER
```

   b. Pouvez-vous effectuer cette opération à l'aide de l'opérateur IN ? Pourquoi ?

```
no rows selected
```

## Exercice 9

3. Ecrivez une requête pour rechercher tous les employés dont le salaire est supérieur au salaire moyen de leur département. Affichez le numéro de chaque employé, son salaire, son numéro de département et le salaire moyen du département. Triez le résultat en fonction du salaire moyen.

| ENAME | SALARY | DEPTNO | DEPT_AVG |
|---|---|---|---|
| ALLEN | 1600 | 30 | 1566.6667 |
| BLAKE | 2850 | 30 | 1566.6667 |
| JONES | 2975 | 20 | 2175 |
| FORD | 3000 | 20 | 2175 |
| SCOTT | 3000 | 20 | 2175 |
| KING | 5000 | 10 | 2916.6667 |

4. Ecrivez une requête pour afficher les employés dont le salaire est inférieur à la moitié du salaire moyen de leur département.

| ENAME |
|---|
| SMITH |
| MILLER |

5. Ecrivez une requête pour afficher les employés ayant un ou plusieurs collègues de leur département dont les dates d'embauche sont postérieures aux leurs et dont les salaires sont plus élevés que les leurs.

| ENAME |
|---|
| CLARK |
| JONES |
| ALLEN |
| WARD |
| SMITH |

2. Créez un état représentant l'organigramme du département de Jones. Imprimez les noms des employés, leur salaire et leur numéro de département.

| ENAME | SAL | DEPTNO |
|---|---|---|
| JONES | 2975 | 20 |
| FORD | 3000 | 20 |
| SMITH | 800 | 20 |
| SCOTT | 3000 | 20 |
| ADAMS | 1100 | 20 |

3. Créez un état dans lequel figurent les noms de tous les responsables pour lesquels travaille Adams.

| ENAME |
|---|
| SCOTT |
| JONES |
| KING |

4. Créez un état représentant la hiérarchie des dirigeants par une indentation. Affichez les noms des employés, le numéro de leur département ainsi que le numéro de leur responsable. Commencez par l'employé ayant le grade le plus élevé.

| NAME | MGR | DEPTNO |
|---|---|---|
| KING | | 10 |
| BLAKE | 7839 | 30 |
| MARTIN | 7698 | 30 |
| ALLEN | 7698 | 30 |
| TURNER | 7698 | 30 |
| JAMES | 7698 | 30 |
| WARD | 7698 | 30 |
| CLARK | 7839 | 10 |
| MILLER | 7782 | 10 |
| JONES | 7839 | 20 |
| FORD | 7566 | 20 |
| SMITH | 7902 | 20 |
| SCOTT | 7566 | 20 |
| ADAMS | 7788 | 20 |

S'il vous reste encore du temps, effectuez l'exercice suivant :

5. Créez l'organigramme d'une société représentant la hiérarchie des dirigeants. Commencez par la personne ayant le grade le plus élevé et excluez tous les employés occupant le poste ANALYST, ainsi que le département de CLARK.

| ENAME | EMPNO | MGR |
|---|---|---|
| KING | 7839 | |
| BLAKE | 7698 | 7839 |
| MARTIN | 7654 | 7698 |
| ALLEN | 7499 | 7698 |
| TURNER | 7844 | 7698 |
| JAMES | 7900 | 7698 |
| WARD | 7521 | 7698 |
| JONES | 7566 | 7839 |
| SMITH | 7369 | 7902 |
| ADAMS | 7876 | 7788 |

# Présentation des Exercices

- Insertion de lignes dans une table.
- Mise à jour et suppression de lignes dans une table.
- Contrôle des transactions.

12-39

### Présentation des Exercices

Au cours des exercices qui suivent, vous allez ajouter des lignes dans la table MY_EMPLOYEE, mettre à jour et supprimer des données de cette table, et contrôler vos transactions.

## Exercices 12

Insérez des données dans la table MY_EMPLOYEE.

1. Exécutez le script \LABS*lab9_ 1.sql* pour créer la table MY_EMPLOYEE qui va servir pour cette série d'exercices.

2. Affichez la structure de la table MY_EMPLOYEE pour trouver les noms de colonnes.

```
Name            NULL?      Type
--------------- ---------- ------
ID              NOT NULL   NUMBER(4)
LAST_NAME                  VARCHAR2(25)
FIRST_NAME                 VARCHAR2(25)
USERID                     VARCHAR2(8)
SALARY                     NUMBER(9,2)
```

3. Ajoutez la première ligne de données du tableau ci-dessous dans la table MY_EMPLOYEE. N'énumérez pas les colonnes dans la clause INSERT.

| ID | LAST_NAME | FIRST_NAME | USERID | SALARY |
|---|---|---|---|---|
| 1 | Patel | Ralph | rpatel | 795 |
| 2 | Dancs | Betty | bdancs | 860 |
| 3 | Biri | Ben | bbiri | 1100 |
| 4 | Newman | Chad | cnewman | 750 |
| 5 | Ropeburn | Audry | aropebur | 1550 |

4. Continuez à remplir la table MY_EMPLOYEE en insérant la seconde ligne des données ci-dessus. Cette fois, mentionnez explicitement les colonnes dans la clause INSERT.

5. Vérifiez vos ajouts.

```
ID  LAST_NAME   FIRST_NAME USERID SALARY
--- ----------- ---------- ------ ------
  1 Patel       Ralph      rpatel    795
  2 Dancs       Betty      bdancs    860
```

## Exercices 12

6. Créez un script nommé *loademp.sql* pour charger les l ignes dans la table MY_EMPLOYEE en mode interactif. Demandez à l'utilisateur de saisir le prénom (first name), le nom (last- name) et le salaire de chaque employé. Concaténez la première lettre du prénom et les sept premières lettres du nom pour former l'ID utilisateur.
7. Insérez les deux lignes de données suivantes dans la table en exécutant le script que vous avez créé.
8. Vérifiez vos ajouts dans la table.

```
 ID LAST_NAME  FIRST_NAME USERID  SALARY
--- ---------- ---------- ------ -------
  1 Patel      Ralph      rpatel     795
  2 Dancs      Betty      bdancs     860
  3 Biri       Ben        bbiri     1100
  4 Newman     Chad       cnewman    750
```

9. Validez vos ajouts.

Mettez à jour et supprimez des données de la table MY_EMPLOYEE.

10. Remplacez le nom de l'employé 3 par Drexler.
11. Saisissez un salaire de 1000 pour tous les employés ayant un salaire inférieur à 900.
12. Vérifiez vos modifications.

```
LAST_NAME SALARY
--------- ------
Patel       1000
Dancs       1000
Biri        1100
Newman      1000
```

13. Supprimez Betty Dancs de la table MY_EMPLOYEE.
14. Vérifiez vos modifications.

```
 ID LAST_NAME  FIRST_NAME USERID SALARY
--- ---------- ---------- ------ ------
  1 Patel      Ralph      rpatel   1000
  3 Drexler    Ben        bbiri    1100
  4 Newman     Chad       cnewman  1000
```

15. Validez toutes les modifications en instance.

    Contrôlez la transaction de données effectuée dans les tables MY_EMPLOYEE.

16. Insérez la dernière ligne des données d'exemple dans la table en exécutant le script que vous avez créé à l'étape 6.

17. Vérifiez l'ajout.

```
        ID LAST_NAME  FIRST_NAME  USERID    SALARY
---------- ---------- ----------- -------- ------
         1 Patel      Ralph       rpatel     1000
         3 Drexler    Ben         bbiri      1100
         4 Newman     Chad        cnewman    1000
         5 Ropeburn   Audry       aropebur   1500
```

18. Définissez une étiquette intermédiaire dans le traitement de la transaction.

19. Videz entièrement la table.

20. Vérifiez que la table est vide.

21. Rejetez la dernière opération DELETE sans annuler l'opération INSERT précédente.

22. Vérifiez que la dernière ligne est restée intacte.

```
        ID LAST_NAME  FIRST_NAME  USERID    SALARY
---------- ---------- ----------- -------- ------
         1Patel       Ralph       rpatel      795
         3Biri        Ben         bbiri      1100
         4Newman      Chad        cnewman     750
         5Ropeburn    Audry       aropebur   1500
```

23. Rendez les ajouts définitifs.

## Exercices 13

1. Créez la table DEPARTMENT d'après le tableau suivant. Saisissez la syntaxe dans un script que vous nommerez *p10q1.sql*, puis exécutez ce script pour créer la table. Vérifiez la création de la table.

| Colonne | Id | Name |
|---|---|---|
| **Type de clé** | | |
| **Null/Unique** | | |
| **Table FK** | | |
| **Colonne FK** | | |
| **Type de données** | Number | Varchar2 |
| **Longueur** | 7 | 25 |

```
Name          NULL?    Type
------------- -------- ----
ID                     NUMBER(7)
NAME                   VARCHAR2(25)
```

2. Remplissez la table DEPARTMENT avec les données de la table DEPT. N'utilisez que les colonnes dont vous avez besoin.

3. Créez la table EMPLOYEE d'après le tableau suivant. Saisissez la syntaxe dans un script que vous nommerez *p10q3.sql*, puis exécutez ce script pour créer la table. Vérifiez la création de la table.

| Colonne | ID | LAST_NAME | FIRST_NAME | DEPT_ID |
|---|---|---|---|---|
| **Type de clé** | | | | |
| **Null/Unique** | | | | |
| **Table FK** | | | | |
| **Colonne FK** | | | | |
| **Type de données** | Number | Varchar2 | Varchar2 | Number |
| **Longueur** | 7 | 25 | 25 | 7 |

```
Name            NULL?    Type
--------------- -------- ----
ID                       NUMBER(7)
LAST_NAME                VARCHAR2(25)
FIRST_NAME               VARCHAR2(25)
DEPT_ID                  NUMBER(7)
```

## Exercices 13

4. Modifiez la table EMPLOYEE pour pouvoir allonger les noms de famille des employés. Vérifiez votre modification.

```
Name            NULL?     Type
--------------- --------- ----
ID                        NUMBER(7)
LAST_NAME                 VARCHAR2(50)
FIRST_NAME                VARCHAR2(25)
DEPT_ID                   NUMBER(7)
```

5. Vérifiez que les tables DEPARTMENT et EMPLOYEE sont bien enregistrées dans le dictionnaire de données. (*Utiliser* : USER_TABLES)

```
TABLE_NAME
------------------------------
DEPARTMENT
EMPLOYEE
```

6. Créez la table EMPLOYEE2 sur la base de la structure de la table EMP et n'incluez que les colonnes EMPNO, ENAME et DEPTNO. Nommez les colonnes de votre nouvelle table respectivement ID, LAST_NAME et DEPT_ID,.
7. Supprimez la table EMPLOYEE.
8. Renommez la table EMPLOYEE2 en EMPLOYEE.
9. Ajoutez un commentaire aux définitions de tables DEPARTMENT et EMPLOYEE pour décrire chaque table. Vérifiez vos ajouts dans le dictionnaire de données.

# Présentation des Exercices

- Ajout de contraintes à des tables existantes
- Ajout de colonnes supplémentaires à une table
- Affichage d'informations des vues du dictionnaire de données

14-25

### Présentation des Exercices

Dans les exercices qui suivent, vous allez ajouter des contraintes et des colonnes supplémentaires dans une table en utilisant les ordres étudiés dans ce chapitre.

## Exercices 14

1. Ajoutez une contrainte PRIMARY KEY de niveau table dans la table EMPLOYEE en utilisant la colonne ID.
   La contrainte devrait être activée dès sa création.
2. Créez une contrainte PRIMARY KEY sur la table DEPARTMENT en utilisant la colonne ID. La contrainte devrait être activée dès sa création.
3. Ajoutez une clé étrangère dans la table EMPLOYEE qui permettra de contrôler que l'employé n'est pas associé à un département inexistant.
4. Vérifiez que les contraintes ont été ajoutées en les recherchant dans USER_CONSTRAINTS. Prenez note des types et des noms des contraintes. Enregistrez votre ordre dans un fichier nommé *p11q4.sql*.

```
CONSTRAINT_NAME              C
---------------------------- --
DEPARTMENT_ID_PK             P
EMPLOYEE_ID_PK               P
EMPLOYEE_DEPT_ID_FK          R
```

5. Recherchez le nom et le type des objets dans la vue USER_OBJECTS du dictionnaire de données correspondant aux tables EMPLOYEE et DEPARTMENT. Vous pouvez mettre les colonnes en forme pour qu'elles soient plus lisibles. Remarquez que pour chaque nouvelle table un nouvel index a été créé.

```
OBJECT_NAME          OBJECT_TYPE
-------------------- -----------
DEPARTMENT           TABLE
DEPARTMENT_ID_PK     INDEX
EMPLOYEE             TABLE
EMPLOYEE_ID_PK       INDEX
```

Si vous avez le temps, faites l'exercice suivant :

6. Modifiez la table EMPLOYEE. Ajoutez une colonne SALARY dont le type de donnée est NUMBER, avec une longueur 7.

## Exercice 15

1. Créez la vue EMP_VU à partir de la table EMP contenant des numéros et des noms d'employés avec leur numéro de département. Modifiez l'en-tête de la colonne des noms d'employés en la nommant EMPLOYEE.

2. Affichez le contenu de la vue EMP_VU.

```
EMPNO EMPLOYEE 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
 7788 SCOTT        20
 7876 ADAMS        20
 7934 MILLER       10
14 rows selected.
```

3. Sélectionnez le nom de la vue (view_name) et le texte correspondant dans la table USER_VIEWSdu dictionnaire de données.

```
VIEW_NAME   TEXT
----------- ------------------------------------
EMP_VU      SELECT empno, ename employee, deptno
            FROM emp
```

4. A partir de votre vue EMP_VU, faites une requête pour afficher tous les noms des employés et le numéro de leur département.

```
EMPLOYEE       DEPTNO
-------------- ------
KING               10
BLAKE              30
CLARK              10
JONES              20
MARTIN             30
...
14 rows selected.
```

## Exercice 15

5. Créez la vue DEPT20 avec les numéros et les noms de tous les employés du département 20. Nommez les colonnes de la vue respectivement EMPLOYEE_ID, EMPLOYEE et DEPARTMENT_ID. Cette vue ne doit pas autoriser l'affectation d'un employé à un autre département.

6. Affichez la structure et le contenu de la vue DEPT20.

```
Name                 Null?    Type
-------------------- -------- ------
EMPLOYEE_ID          NOT NULL NUMBER(4)
EMPLOYEE                      VARCHAR2(10)
DEPARTMENT_ID        NOT NULL NUMBER(2)
```

```
EMPLOYEE_ID EMPLOYEE DEPARTMENT_ID
----------- -------- -------------
       7566 JONES                20
       7902 FORD                 20
       7369 SMITH                20
       7788 SCOTT                20
       7876 ADAMS                20
```

7. Essayez d'affecter l'employé Smith au département 30.

S'il vous reste du temps, effectuez l'exercice suivant.

8. Créez la vue SALARY_VU de façon à afficher le nom de tous les employés, le nom de leur département, leur salaire et leur barème de salaire. Nommez les colonnes respectivement Employee, Department, Salary et Grade.

# Présentation des Exercices

- Création de séquences
- Utilisation des séquences
- Création d'index non uniques
- Affichage des informations du dictionnaire de données relatives aux séquences et aux index
- Suppression d'index

16-25

### Présentation des Exercices

Dans ces exercices, vous allez créer une séquence pour remplir votre table DEPARTMENT, ainsi que des index implicites et explicites.

## Exercice 16

1. Créez une séquence pour l'utiliser avec la clé primaire de la table DEPARTMENT. Cette séquence doit commencer avec le numéro 60 et avoir une valeur maximale de 200. Attribuez un pas d'incrémentation de 10 et nommez la séquence DEPT_ID_SEQ.
2. Ecrivez un script pour afficher les informations suivantes relatives à vos séquences : nom de la séquence, valeur maximale, pas d'incrémentation et dernier numéro de séquence. Nommez le script *p13q2.sql* puis exécutez-le.

```
SEQUENCE_NAME MAX_VALUE INCREMENT_BY LAST_NUMBER
------------- --------- ------------ -----------
CUSTID        1.000E+27             1         109
DEPT_ID_SEQ         200             1          60
ORDID         1.000E+27             1         622
PRODID        1.000E+27             1      200381
```

3. Ecrivez un script interactif pour insérer une ligne dans la table DEPARTMENT. Nommez votre script *p13q3.sql*. Veillez à utiliser la séquence que vous avez créé pour la colonne ID. Créez une invite personnalisée pour la saisie du nom du département. Exécutez votre script. Ajoutez les deux départements Education et Administration, puis validez-les.
4. Créez un index non unique dans la colonne FOREIGN KEY de la table EMPLOYEE.
5. Affichez les index et l'unicité en utilisant le dictionnaire de données pour la table EMPLOYEE. Enregistrez l'ordre dans le script *p13q5.sql*.

```
INDEX_NAME                TABLE_NAME   UNIQUENES
------------------------- ------------ ---------
EMPLOYEE_DEPT_ID_IDX      EMPLOYEE     NONUNIQUE
EMPLOYEE_ID_PK            EMPLOYEE     UNIQUE
```

## Exercice 17

1. Quel privilège doit avoir un utilisateur pour se connecter à Oracle8 Server ? S'agit-il d'un privilège système ou objet ?

   _____

2. Quel privilège doit avoir un utilisateur pour créer des tables ?

   _____

3. Si vous créez une table, qui peut transmettre à d'autres utilisateurs les privilèges liés à votre table ?

   _____

4. Vous êtes administrateur de base de données et vous devez créer un grand nombre d'utilisateurs qui exigent les mêmes privilèges système. Comment pouvez-vous simplifier cette tâche ?

   _____

5. Quelle commande pouvez-vous utiliser pour modifier votre mot de passe ?

   _____

6. Autorisez un autre utilisateur à lire votre table DEPT. Demandez à un utilisateur de vous accorder un droit de lecture sur sa table DEPT.
7. Interrogez toutes les lignes de votre table DEPT.

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

8. Ajoutez une nouvelle ligne à votre table DEPT. L'équipe 1 doit ajouter le département Education portant le numéro 50, et l'équipe 2 le département Administration portant le numéro 50. Enregistrez vos modifications.
9. Créez un synonyme pour la table DEPT de l'autre équipe.