

3

Fonctions monoligne

ORACLE®

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Objectifs

A la fin de ce chapitre, vous pourrez :

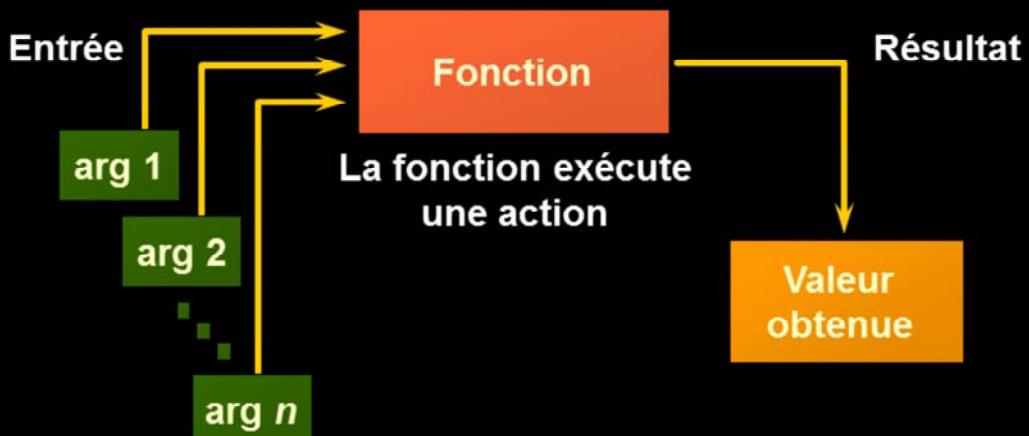
- **décrire différents types de fonction SQL**
- **utiliser des fonctions alphanumériques, numériques et de date dans les instructions SELECT**
- **décrire l'utilisation des fonctions de conversion**

ORACLE

But du chapitre

Les **fonctions** augmentent la puissance du bloc d'interrogation de base et permettent de manipuler des valeurs de données. Les fonctions sont abordées dans deux chapitres, dont celui-ci est le premier. Il présente en particulier les fonctions monoligne alphanumériques, numériques et de date, ainsi que les fonctions de conversion des types de données (qui permettent, par exemple, de convertir des données alphanumériques en données numériques).

Fonctions SQL



ORACLE

3-3

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions SQL

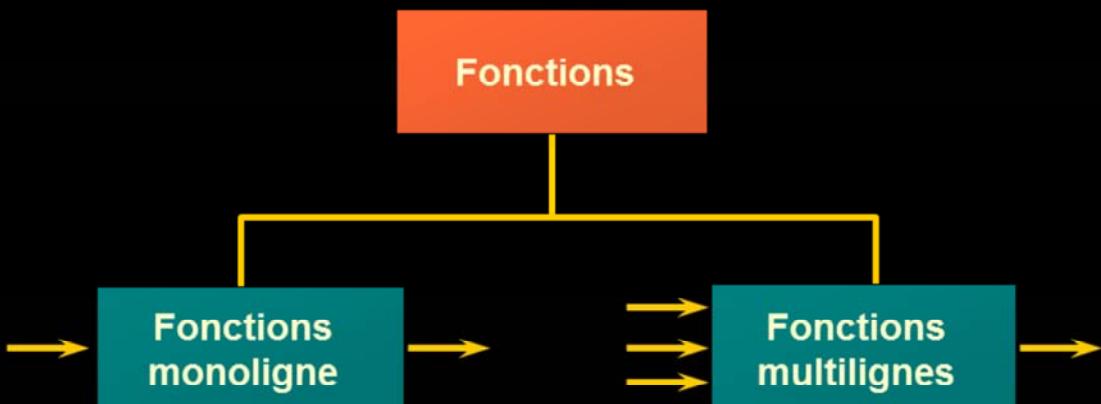
Les **fonctions** constituent une entité extrêmement puissante de SQL permettant :

- d'effectuer des calculs sur des données,
- de modifier des éléments de données individuels,
- de manipuler le résultat de groupes de lignes,
- de formater des dates et des nombres pour affichage,
- de convertir les types de données des colonnes.

Les fonctions SQL utilisent parfois des **arguments** et **renvoient toujours une valeur**.

Remarque : La plupart des fonctions SQL décrites dans ce chapitre sont spécifiques à la version SQL d'Oracle.

Deux types de fonction SQL



3-4

Copyright © Oracle Corporation, 2001. Tous droits réservés.

ORACLE

Fonctions SQL (suite)

On distingue deux types de fonction :

- les fonctions monoligne,
- les fonctions multilignes.

Fonctions monoligne

Ces fonctions agissent sur une seule ligne à la fois et renvoient un résultat par ligne. Parmi les types de fonction monoligne existants, ce chapitre aborde les quatre suivants :

- Fonction alphanumérique
- Fonction numérique
- Fonction de date
- Fonction de conversion

Fonctions multilignes

Ces fonctions, également appelées fonctions de groupe, peuvent manipuler des groupes de lignes et renvoyer un résultat par groupe. Elles sont présentées dans un chapitre ultérieur.

Pour obtenir la syntaxe et la liste complète des fonctions disponibles, voir *Oracle9i SQL Reference*.

Fonctions monoligne

Les fonctions monoligne :

- manipulent des éléments de données,
- acceptent des arguments et renvoient une valeur,
- agissent sur chacune des lignes renvoyées,
- renvoient un résultat par ligne,
- peuvent modifier le type de données,
- peuvent être imbriquées,
- acceptent des arguments qui peuvent être une colonne ou une expression.

```
function_name [(arg1, arg2,...)]
```

ORACLE

3-5

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions monoligne

Les fonctions monoligne permettent de manipuler des éléments de données. Elles acceptent un ou plusieurs arguments et renvoient une valeur pour chaque ligne renvoyée par l'interrogation. Les éléments suivants peuvent être des arguments :

- Constante fournie par l'utilisateur
- Valeur de variable
- Nom de colonne
- Expression

Caractéristiques des fonctions monoligne :

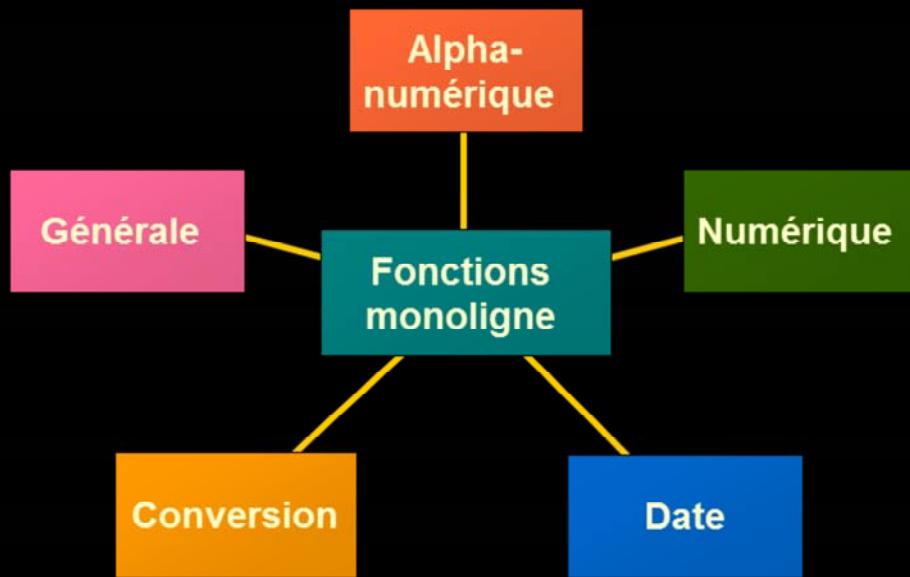
- Agissent sur chaque ligne renvoyée par l'interrogation.
- Renvoient un résultat par ligne.
- Peuvent renvoyer une valeur de données dont le type diffère du type référencé.
- Acceptent un ou plusieurs arguments.
- Peuvent être imbriquées et utilisées dans les clauses SELECT, WHERE et ORDER BY.

Explication de la syntaxe :

function_name désigne le nom de la fonction.

arg1, arg2 désigne tout argument que la fonction doit utiliser. Peut être représentée par un nom de colonne ou une expression.

Fonctions monoligne



ORACLE

3-6

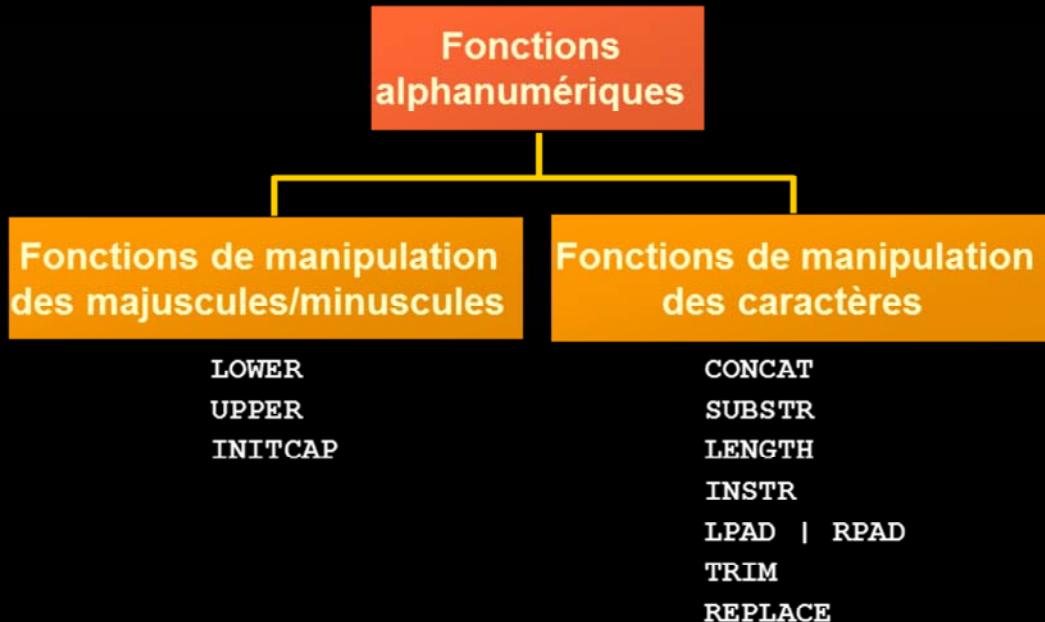
Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions monoligne (suite)

Ce chapitre présente les fonctions monoligne suivantes :

- Fonctions alphanumériques – Acceptent des caractères en entrée et peuvent renvoyer des valeurs alphanumériques ou numériques.
- Fonctions numériques – Acceptent des valeurs numériques en entrée et renvoient des valeurs numériques.
- Fonctions de date – Opèrent sur des **valeurs de type DATE** (toutes les fonctions de date renvoient des valeurs de type DATE, excepté la fonction **MONTHS_BETWEEN** qui renvoie une valeur numérique).
- Fonctions de conversion – Convertissent une valeur d'un type de données en un autre.
- Fonctions générales :
 - **NVL**
 - **NVL2**
 - **NULLIF**
 - **COALESCE**
 - **CASE**
 - **DECODE**

Fonctions alphanumériques



ORACLE

3-7

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions alphanumériques

Les fonctions monoligne alphanumériques acceptent des données alphanumériques en entrée et peuvent renvoyer des données alphanumériques ou numériques. Elles se divisent en deux catégories :

- Fonctions de manipulation des majuscules/minuscules
- Fonctions de manipulation des caractères

Fonction	Opération
LOWER(<i>column/expression</i>)	Convertit des valeurs alphanumériques en minuscules.
UPPER(<i>column/expression</i>)	Convertit des valeurs alphanumériques en majuscules.
INITCAP(<i>column/expression</i>)	Convertit la première lettre d'une valeur alphanumérique en majuscules et les autres lettres en minuscules.
CONCAT(<i>column1/expression1, column2/expression2</i>)	Concatène la première valeur alphanumérique à la deuxième. Correspond à l'opérateur de concaténation ().
SUBSTR(<i>column/expression, m [,n]</i>)	Renvoie des caractères donnés de la valeur alphanumérique en commençant à la position <i>m</i> et sur <i>n</i> caractères (si <i>m</i> est négatif, le décompte commence à la fin de la valeur alphanumérique. Si <i>n</i> n'est pas indiqué, tous les caractères jusqu'à la fin de la chaîne sont renvoyés).

Remarque : Seule une partie des fonctions disponibles sont présentées dans ce chapitre.

Fonctions alphanumériques

Fonctions alphanumériques

Fonctions de manipulation des majuscules/minuscules

LOWER
UPPER
INITCAP

Fonctions de manipulation des caractères

CONCAT
SUBSTR
LENGTH
INSTR
LPAD | RPAD
TRIM
REPLACE

ORACLE

3-8

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions alphanumériques (suite)

Fonction	Opération
LENGTH(<i>column/expression</i>)	Renvoie le nombre de caractères de l'expression.
INSTR(<i>column/expression</i> , ' <i>string</i> ', [, <i>m</i>], [<i>n</i>])	Renvoie la position numérique d'une chaîne indiquée. Vous pouvez éventuellement indiquer la position <i>m</i> de début de la recherche et l'occurrence <i>n</i> de la chaîne. <i>m</i> et <i>n</i> prennent par défaut la valeur 1, ce qui signifie que la recherche commence au début de la chaîne et renvoie la première occurrence.
LPAD(<i>column/expression</i> , <i>n</i> , ' <i>string</i> ') RPAD(<i>column/expression</i> , <i>n</i> , ' <i>string</i> ')	Ajoute des caractères de remplissage à la valeur alphanumérique en partant de la droite, sur un espace équivalent à <i>n</i> caractères. Ajoute des caractères de remplissage à la valeur alphanumérique en partant de la gauche, sur un espace équivalent à <i>n</i> caractères.
TRIM(<i>leading/trailing/both</i> , <i>trim_character FROM</i> <i>trim_source</i>)	Permet de retirer les caractères de tête ou de fin (ou les deux) d'une chaîne de caractères. Si <i>trim_character</i> ou <i>trim_source</i> est un littéral de type caractère, vous devez le placer entre apostrophes. Cette fonction est disponible dans Oracle8i et les versions ultérieures.
REPLACE(<i>text</i> , <i>search_string</i> , <i>replacement_string</i>)	Recherche une chaîne de caractères dans une expression de type texte et, s'il la trouve, la remplace par une chaîne donnée.

Fonctions de manipulation des majuscules/minuscules

Ces fonctions permettent de modifier la casse des caractères dans des chaînes.

Fonction	Résultat
LOWER ('SQL Course')	sql course
UPPER ('SQL Course')	SQL COURSE
INITCAP ('SQL Course')	Sql Course

ORACLE

3-9

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de manipulation des majuscules/minuscules

Les trois fonctions de conversion majuscules/minuscules sont **LOWER**, **UPPER** et **INITCAP**.

- 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.

```
SELECT 'The job id for '||UPPER(last_name)||' is '||LOWER(job_id) AS "EMPLOYEE DETAILS"
```

EMPLOYEE DETAILS
The job id for KING is ad_pres
The job id for KOCHHAR is ad_vp
The job id for DE HAAN is ad_vp
...
The job id for HIGGINS is ac_mgr
The job id for GIETZ is ac_account

20 rows selected.

Utiliser les fonctions de manipulation des majuscules/minuscules

Affichez le numéro, le nom et le numéro de service de l'employé Higgins :

```
SELECT employee_id, last_name, department_id  
FROM   employees  
WHERE  last_name = 'higgins';  
no rows selected
```

```
SELECT employee_id, last_name, department_id  
FROM   employees  
WHERE  LOWER(last_name) = 'higgins';
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
205	Higgins	110

ORACLE

3-10

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de manipulation des majuscules/minuscules (suite)

L'exemple de la diapositive permet d'afficher le numéro, le nom et le numéro de service de l'employé Higgins.

La clause **WHERE** de la première instruction SQL indique le nom de l'employé sous la forme higgins. Puisque les données de la table EMPLOYEES sont stockées en majuscules, le nom higgins n'est pas identifié dans la table et aucune ligne n'est sélectionnée.

La clause WHERE de la seconde instruction SQL indique que la colonne LAST_NAME doit être convertie en minuscules pour permettre la comparaison du nom d'employé figurant dans la table EMPLOYEES avec higgins. Les deux noms étant désormais en minuscules, la ligne correspondante est sélectionnée. La clause WHERE, réécrite de la manière suivante, produit le même résultat :

```
... WHERE last_name = 'Higgins'
```

Le nom de l'employé s'affiche tel qu'il est stocké dans la base de données. Pour l'afficher en majuscules, utilisez la fonction **UPPER** dans l'instruction SELECT.

```
SELECT employee_id, UPPER(last_name), department_id  
FROM   employees  
WHERE  INITCAP(last_name) = 'Higgins';
```

Fonctions de manipulation des caractères

Ces fonctions permettent de manipuler des chaînes de caractères :

Fonction	Résultat
<code>CONCAT('Hello', 'World')</code>	HelloWorld
<code>SUBSTR('HelloWorld',1,5)</code>	Hello
<code>LENGTH('HelloWorld')</code>	10
<code>INSTR('HelloWorld', 'W')</code>	6
<code>LPAD(salary,10,'*')</code>	*****24000
<code>RPAD(salary, 10, '*')</code>	24000*****
<code>TRIM('H' FROM 'HelloWorld')</code>	elloWorld

ORACLE

3-11

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de manipulation des caractères

Les fonctions de manipulation des caractères `CONCAT`, `SUBSTR`, `LENGTH`, `INSTR`, `LPAD`, `RPAD` et `TRIM` sont présentées dans ce chapitre.

- `CONCAT` : Concatène des valeurs (cette fonction n'accepte que deux paramètres).
- `SUBSTR` : Extrait une chaîne d'une longueur déterminée.
- `LENGTH` : Fournit une valeur numérique correspondant au nombre de caractères d'une chaîne.
- `INSTR` : Fournit une valeur numérique correspondant à la position d'un caractère.
- `LPAD` : Ajoute des caractères de remplissage à une valeur alphanumérique en partant de la droite.
- `RPAD` : Ajoute des caractères de remplissage à une valeur alphanumérique en partant de la gauche.
- `TRIM` : Retire les caractères de tête ou de fin (ou les deux) d'une chaîne de caractères (si `trim_character` ou `trim_source` est un littéral de type caractère, vous devez le placer entre apostrophes).

Utiliser les fonctions de manipulation des caractères

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH(last_name),
       INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(job_id, 4) = 'REP';
    
```

EMPLOYEE_ID	NAME	JOB_ID	LENGTH(LAST_NAME)	Contains 'a'?
174	EllenAbel	SA_REP	4	0
176	JonathonTaylor	SA_REP	6	2
178	KimberelyGrant	SA_REP	5	3
202	PatFay	MK_REP	3	2

1

2

3

ORACLE

3-12

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de manipulation des caractères (suite)

L'exemple de la diapositive permet d'afficher les nom et prénom concaténés des employés, la longueur du nom des employés et la position de la lettre *a* dans le nom des employés dont l'ID de poste comporte en quatrième position la chaîne REP.

Exemple

Modifiez l'instruction SQL de la diapositive pour afficher les données des employés dont le nom se termine par la lettre *n*.

```

SELECT employee_id, CONCAT(first_name, last_name) NAME,
       LENGTH(last_name), INSTR(last_name, 'a') "Contains 'a'?"
FROM   employees
WHERE  SUBSTR(last_name, -1, 1) = 'n';
    
```

EMPLOYEE_ID	NAME	LENGTH(LAST_NAME)	Contains 'a'?
102	LexDe Haan	7	5
200	JenniferWhalen	6	3
201	MichaelHartstein	9	2

Fonctions numériques

- **ROUND** : Arroindit la valeur à la décimale spécifiée

ROUND(45.926, 2)  45.93

- **TRUNC** : Tronque la valeur à la décimale spécifiée

TRUNC(45.926, 2)  45.92

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

MOD(1600, 300)  100

ORACLE

3-13

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions numériques

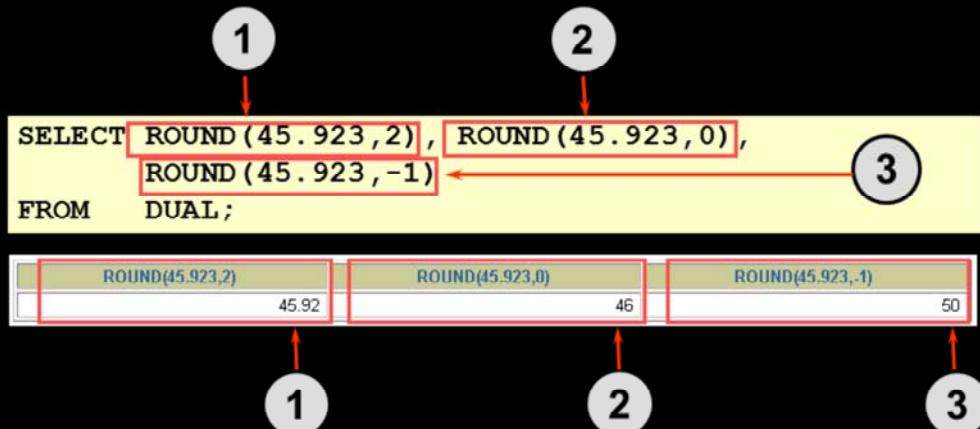
Les **fonctions numériques** acceptent et renvoient des valeurs numériques. Cette section décrit quelques-unes de ces fonctions.

Fonction	Opération
ROUND(<i>column expression, n</i>)	Arrondit la colonne, l'expression ou la valeur à <i>n</i> décimales ou à 0 décimale si <i>n</i> n'est pas indiqué. (Si <i>n</i> est négatif les chiffres situés à gauche du signe décimal sont arrondis.)
TRUNC(<i>column expression, n</i>)	Tronque la colonne, l'expression ou la valeur à <i>n</i> décimales. Si <i>n</i> n'est pas indiqué, il prend par défaut la valeur zéro.
MOD(<i>m, n</i>)	Renvoie le reste de la division de <i>m</i> par <i>n</i> .

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

Pour plus d'informations, voir *Oracle9i SQL Reference*, "Number Functions".

Utiliser la fonction ROUND



DUAL est une table factice dans laquelle vous pouvez visualiser les résultats des fonctions et des calculs.

ORACLE

3-14

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction ROUND

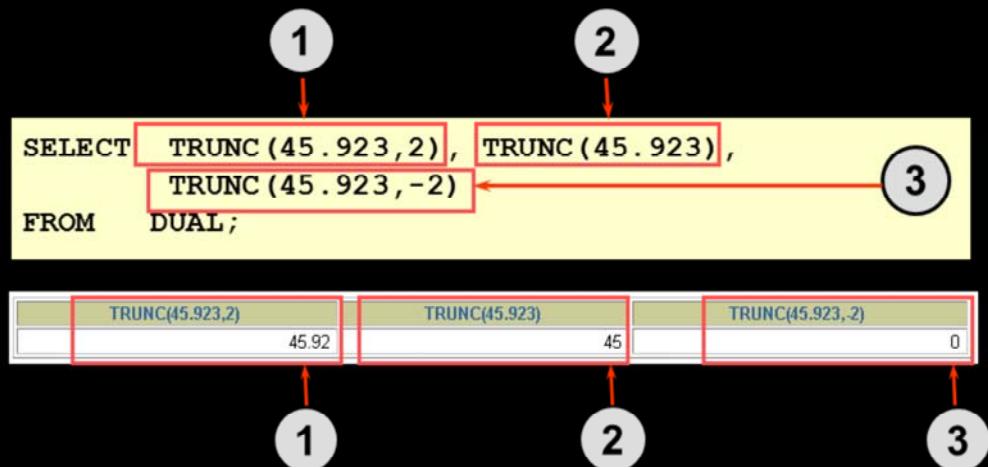
La **fonction ROUND** arrondit la colonne, l'expression ou la valeur à n décimales. Si le deuxième argument vaut 0 ou est absent, la valeur est arrondie à zéro décimale. S'il vaut 2, la valeur est arrondie à deux décimales. Inversement, si le deuxième argument vaut -2, la valeur est arrondie à deux chiffres à gauche de la virgule.

Vous pouvez également utiliser la fonction ROUND avec des fonctions de date. Quelques exemples sont présentés plus loin dans ce chapitre.

Table DUAL

La **table DUAL**, accessible à tous les utilisateurs, appartient à l'utilisateur SYS. Elle contient une colonne, DUMMY, et une ligne qui possède la valeur X. Cette table s'avère utile lorsque vous souhaitez ne renvoyer qu'une fois une valeur, par exemple, la valeur d'une constante, d'une pseudo-colonne ou d'une expression qui n'est pas dérivée d'une table contenant des données utilisateur. Elle est généralement utilisée pour garantir que la syntaxe de la clause SELECT est complète, car SELECT et FROM sont obligatoires et de nombreux calculs ne nécessitent pas de sélection dans des tables réelles.

Utiliser la fonction TRUNC



ORACLE

3-15

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction TRUNC

La **fonction TRUNC** tranque la colonne, l'expression ou la valeur à n décimales.

Elle utilise des arguments similaires à ceux de la fonction ROUND. Si le deuxième argument vaut 0 ou est absent, la valeur ne conserve aucune décimale. S'il vaut 2, la valeur est conservée deux décimales. Inversement, si le deuxième argument vaut -2, la valeur conserve deux chiffres à gauche de la virgule. Tout comme la fonction ROUND, TRUNC peut être utilisée avec des fonctions de date.

Utiliser la fonction MOD

Calculez le reste d'un salaire après division par 5000 pour tous les employés occupant le poste de représentant.

```
SELECT last_name, salary, MOD(salary, 5000)  
FROM   employees  
WHERE  job_id = 'SA_REP';
```

LAST_NAME	SALARY	MOD(SALARY,5000)
Abel	11000	1000
Taylor	8600	3600
Grant	7000	2000

ORACLE

Fonction MOD

La **fonction MOD** calcule le reste de la division d'une valeur1 par une valeur2. L'exemple de la diapositive permet de calculer le reste de la division du salaire par 5000, pour tous les employés dont l'ID de poste est SA_REP.

Remarque : Cette fonction est généralement utilisée pour déterminer si une valeur est paire ou impaire.

Utiliser les dates

- Dans la base de données Oracle, les dates sont stockées sous un format numérique interne : siècle, année, mois, jour, heures, minutes, secondes.
- Le format de date par défaut est DD-MON-RR.
 - Vous pouvez ainsi enregistrer des dates du 21e siècle au 20e siècle en indiquant uniquement les deux derniers chiffres de l'année.
 - Inversement, vous pouvez enregistrer des dates du 20e siècle au 21e siècle.

```
SELECT last_name, hire_date  
FROM employees  
WHERE last_name like 'G%';
```

LAST_NAME	HIRE_DATE
Gietz	07-JUN-94
Grant	24-MAY-99

ORACLE

3-17

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Format de date Oracle

Dans la base de données Oracle, les **dates** sont stockées sous 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-RR. Les dates valides pour Oracle sont comprises entre le 1^{er} janvier 4712 av. J.-C. et le 31 décembre 9999 apr. J.-C.

Dans l'exemple de la diapositive, la date d'embauche (HIRE_DATE) de l'employé Gietz s'affiche par défaut au format DD-MON-RR. Cependant, les dates ne sont pas stockées sous ce format dans la base de données. Tous les composants de date et d'heure sont stockés. Ainsi, bien qu'une date d'embauche telle que le 7 juin 94 indique le jour, le mois et l'année, des informations relatives à l'*heure* et au *siècle* y sont également associées. L'information complète est de type : 7 juin 1994 17:10:43.

En interne, les données sont stockées de la façon suivante :

SIECLE	ANNEE	MOIS	JOUR	HEURE	MINUTE	SECONDE
19	94	06	07	5	10	43

Prise en charge des siècles et de l'an 2000

Le serveur Oracle est **conforme An 2000**. Lorsqu'un enregistrement comportant une colonne de date est inséré dans une table, les informations relatives au *siècle* sont obtenues via la fonction SYSDATE. En revanche, lorsque la colonne de date s'affiche à l'écran, le siècle ne s'affiche pas par défaut.

Le type de données DATE enregistre toujours en interne des informations relatives à l'année sous la forme de quatre chiffres : deux pour le siècle et deux pour l'année. Par exemple, la base de données Oracle enregistre l'année sous la forme 1996 ou 2001 et non 96 ou 01.

Utiliser les dates

La fonction SYSDATE renvoie :

- la date,
- l'heure.

ORACLE

3-18

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction SYSDATE

SYSDATE est une fonction de date qui renvoie la date et l'heure du serveur de bases de données. Elle s'utilise de la même façon que n'importe quel autre nom de colonne. Par exemple, vous pouvez sélectionner SYSDATE dans une table pour afficher la date du jour. Il est usuel d'interroger la table factice **DUAL**.

Exemple

Affichez la date du jour au moyen de la table DUAL.

```
SELECT SYSDATE  
FROM   DUAL;
```

SYSDATE
28-SEP-01

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.

ORACLE

3-19

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Opérations arithmétiques sur les dates

Etant donné que les dates sont stockées dans la base sous la forme de données numériques, il est possible d'effectuer des calculs tels que des additions ou des soustractions au moyen d'opérateurs arithmétiques. Vous pouvez ajouter et soustraire des constantes numériques aussi bien que des dates.

Les opérations possibles sont les suivantes :

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

Utiliser des opérateurs arithmétiques avec les dates

```
SELECT last_name, (SYSDATE-hire_date)/7 AS WEEKS  
FROM   employees  
WHERE  department_id = 90;
```

LAST_NAME	WEEKS
King	744.245395
Kochhar	626.102538
De Haan	453.245395

ORACLE

3-20

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Opérations arithmétiques sur les dates (suite)

L'exemple de la diapositive permet d'afficher le nom et le nombre de semaines d'ancienneté de tous les employés du service 90. La date d'embauche de l'employé est soustraite de la date du jour (SYSDATE), puis le résultat est divisé par 7 pour obtenir le nombre de semaines d'ancienneté.

Remarque : La fonction SQL **SYSDATE** renvoie l'heure et la date actuelles. Vos résultats peuvent donc différer de ceux de l'exemple.

La soustraction d'une date à une date antérieure aboutit à un résultat négatif.

Fonctions de date

Fonction	Description
MONTHS_BETWEEN	Nombre de mois entre deux dates
ADD_MONTHS	Ajout de mois calendaires à une date
NEXT_DAY	Jour suivant la date indiquée
LAST_DAY	Dernier jour du mois
ROUND	Arrondi d'une date
TRUNC	Troncature d'une date

ORACLE

3-21

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de date

Les fonctions de date s'appliquent aux dates Oracle. Toutes les fonctions de date renvoient une valeur de type DATE excepté **MONTHS_BETWEEN**, qui renvoie une valeur numérique.

- **MONTHS_BETWEEN(date1, date2)** – Recherche le nombre de mois entre deux dates (*date1* et *date2*). Le résultat peut être positif ou négatif. Si la date *date1* est postérieure à la date *date2*, le résultat est positif, si elle est antérieure, 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')** – Recherche la date du prochain jour indiqué ('*char*') après la date indiquée (*date*). *char* peut être un numéro représentant un jour ou une chaîne de caractères.
- **LAST_DAY(date)** – Recherche la date du dernier jour du mois auquel appartient la date indiquée (*date*).
- **ROUND(date[, 'fmt'])** – Renvoie la date arrondie à l'unité précisée par le modèle de format *fmt*. Si *fmt* n'est pas indiqué, la date est arrondie au jour le plus proche.
- **TRUNC(date[, 'fmt'])** – Renvoie la date (et la partie horaire) tronquée à l'unité de jour précisée par le modèle de format *fmt*. Lorsque *fmt* n'est pas indiqué, la date est tronquée au jour le plus proche.

Cette liste de fonctions de date n'est pas exhaustive. Les modèles de format sont présentés plus loin dans ce chapitre. Le mois ou l'année, par exemple, sont des modèles de format.

Utiliser les fonctions de 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-FEB-95')** → '28-FEB-95'

ORACLE

3-22

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de date (suite)

Exemple : Affichez pour tous les employés ayant moins de 36 mois d'ancienneté le numéro d'employé, 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.

```
SELECT employee_id, hire_date,
       MONTHS_BETWEEN (SYSDATE, hire_date) TENURE,
       ADD_MONTHS (hire_date, 6) REVIEW,
       NEXT_DAY (hire_date, 'FRIDAY'), LAST_DAY(hire_date)
  FROM employees
 WHERE MONTHS_BETWEEN (SYSDATE, hire_date) < 36;
```

EMPLOYEE_ID	HIRE_DATE	TENURE	REVIEW	NEXT_DAY	LAST_DAY
107	07-FEB-99	31.6982407	07-AUG-99	12-FEB-99	28-FEB-99
124	16-NOV-99	22.4079182	16-MAY-00	19-NOV-99	30-NOV-99
149	29-JAN-00	19.9885633	29-JUL-00	04-FEB-00	31-JAN-00
178	24-MAY-99	28.1498536	24-NOV-99	28-MAY-99	31-MAY-99

Utiliser les fonctions de date

Supposons que SYSDATE = '25-JUL-95' :

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

ORACLE

3-23

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de date (suite)

Les fonctions **ROUND** et **TRUNC** peuvent être utilisées avec des valeurs de type numérique ou date. Utilisées avec des dates, ces fonctions procèdent à un arrondi ou à une troncature selon le modèle de format indiqué. Vous pouvez par conséquent arrondir les dates au mois ou à l'année le/la plus proche.

Exemple

Comparez les dates d'embauche de tous les employés recrutés en 1997. Affichez le numéro d'employé, la date d'embauche et le mois de début d'activité à l'aide des fonctions **ROUND** et **TRUNC**.

```
SELECT employee_id, hire_date,
       ROUND(hire_date, 'MONTH'), TRUNC(hire_date, 'MONTH')
  FROM employees
 WHERE hire_date LIKE '%97';
```

EMPLOYEE_ID	HIRE_DATE	ROUND(HIR)	TRUNC(HIR)
142	29-JAN-97	01-FEB-97	01-JAN-97
202	17-AUG-97	01-SEP-97	01-AUG-97

Présentation de l'exercice 3, 1^{ère} partie

Dans cet exercice, vous allez :

- écrire une interrogation permettant d'afficher la date du jour
- créer des interrogations utilisant des fonctions numériques, alphanumériques et de date
- effectuer des calculs relatifs aux années et aux mois d'ancienneté d'un employé

ORACLE

3-24

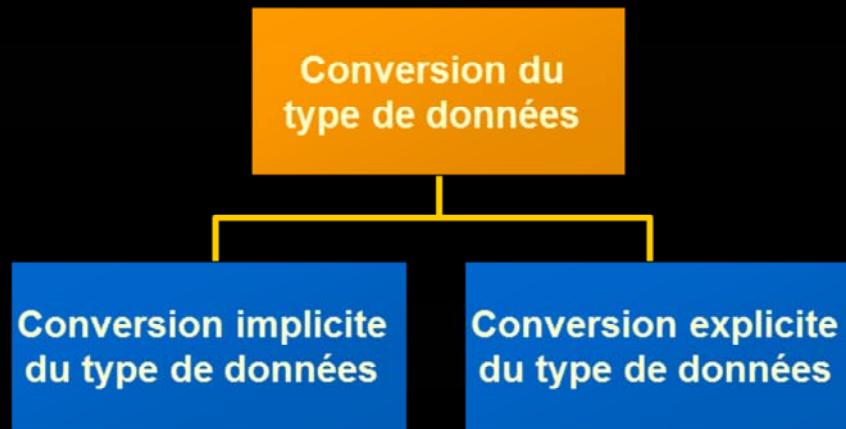
Copyright © Oracle Corporation, 2001. Tous droits réservés.

Présentation de l'exercice 3, 1^{ère} partie

La variété des exercices va vous permettre d'appliquer les différentes fonctions utilisables avec des données de type alphanumérique, numérique et date.

Répondez aux questions 1 à 5 à la fin de ce chapitre.

Fonctions de conversion



ORACLE

3-25

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions de conversion

Outre les types de données Oracle, il est possible d'utiliser les **types de données** ANSI, DB2 et SQL/DS pour définir les colonnes des tables dans une base de données Oracle9i. Cependant, le serveur Oracle convertit en interne ces types de données en types de données Oracle.

Dans certains cas, le serveur Oracle accepte des données d'un type différent de celui normalement attendu. Dans ce cas, il peut automatiquement convertir ces données. Cette conversion des types de données est réalisée *implicitement* par le serveur Oracle ou *explicitement* par l'utilisateur.

Les conversions implicites des types de données fonctionnent selon des règles qui sont présentées dans les deux prochaines diapositives.

Les conversions explicites des types de données s'effectuent à l'aide des fonctions de conversion, qui convertissent une valeur d'un type en un autre. En principe, le format des noms de fonction suit la convention *data type TO data type*. Le premier type de données correspond au type de données d'entrée et le second au type de données de sortie.

Remarque : Bien que la **conversion implicite des types de données** soit possible, il est recommandé d'effectuer une **conversion explicite** afin de garantir la fiabilité des instructions SQL.

Conversion implicite des types de données

Pour les affectations, le serveur Oracle peut convertir automatiquement les types de données suivants :

Type de données source	Type de données cible
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

ORACLE

3-26

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Conversion implicite des types de données

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

Conversion implicite des types de données

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

Type de données source	Type de données cible
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

ORACLE

3-27

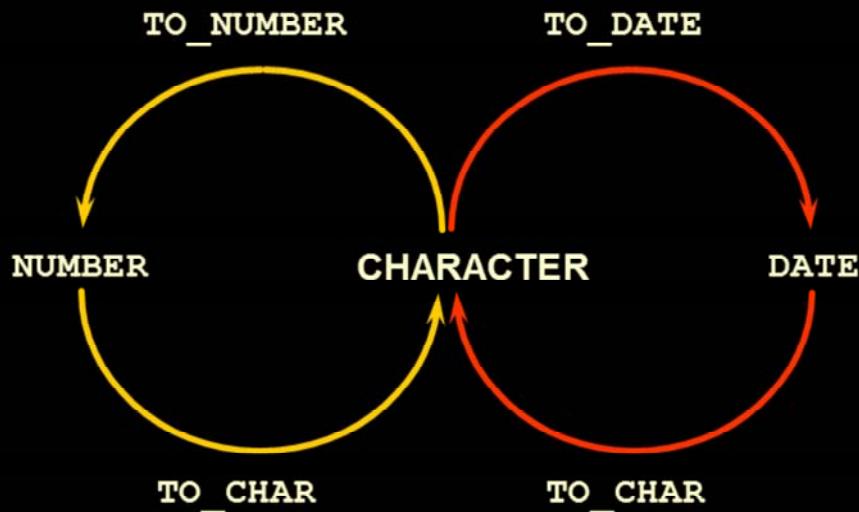
Copyright © Oracle Corporation, 2001. Tous droits réservés.

Conversion implicite des types de données (suite)

En général, le serveur Oracle utilise la règle applicable aux expressions lorsqu'une conversion de type de données est requise à des emplacements qui ne sont régis par aucune règle de conversion des affectations.

Remarque : Les conversions du type CHAR au type NUMBER n'aboutissent que si la chaîne de caractères représente un nombre valide.

Conversion explicite des types de données



ORACLE

3-28

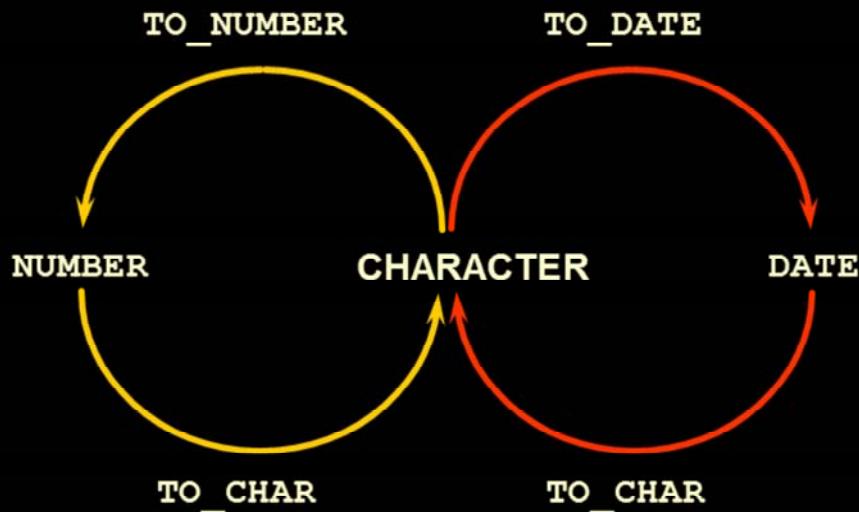
Copyright © Oracle Corporation, 2001. Tous droits réservés.

Conversion explicite des types de données

SQL offre trois fonctions permettant de convertir une valeur d'un type de données dans autre type :

Fonction	Opération
<code>TO_CHAR(number date, [fmt], [nlsparams])</code>	<p>Convertit un nombre ou une date en une chaîne de caractères VARCHAR2 selon le modèle de format <i>fmt</i>.</p> <p>Conversion numérique – Le paramètre <i>nlsparams</i> indique les caractères suivants, qui sont renvoyés par les éléments de format numérique :</p> <ul style="list-style-type: none">• Caractère décimal• Séparateur de groupe• Symbole de devise locale• Symbole de devise internationale <p>En l'absence de <i>nlsparams</i> ou de tout autre paramètre, la fonction utilise les valeurs de paramètre par défaut pour la session.</p>

Conversion explicite des types de données



ORACLE

3-29

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Conversion explicite des types de données (suite)

Fonction	Opération
<code>TO_CHAR(number date, [fmt], [nlsparams])</code>	Conversion de date – Le paramètre <code>nlsparams</code> indique dans quel langage les noms de mois et de jours et les abréviations sont renvoyés. S'il est omis, la fonction utilise les langages de date par défaut pour la session.
<code>TO_NUMBER(char, [fmt], [nlsparams])</code>	Convertit une chaîne de caractères contenant des chiffres en nombre, en respectant le format indiqué dans le modèle de format facultatif <code>fmt</code> . Le paramètre <code>nlsparams</code> joue le même rôle que dans la fonction <code>TO_CHAR</code> de conversion numérique.
<code>TO_DATE(char, [fmt], [nlsparams])</code>	Convertit une chaîne de caractères représentant une date en une valeur de date, en respectant le modèle de format <code>fmt</code> indiqué. Si <code>fmt</code> n'est pas indiqué, le format DD-MON-YY est appliqué. Le paramètre <code>nlsparams</code> joue le même rôle que dans la fonction <code>TO_CHAR</code> de conversion de date.

Conversion explicite des types de données (suite)

Remarque : La liste de fonctions de conversion fournie dans ce chapitre n'est pas exhaustive.

Pour plus d'informations, voir *Oracle9i SQL Reference*, "Conversion Functions".

Utiliser la fonction TO_CHAR avec les dates

```
TO_CHAR(date, 'format_model')
```

Le modèle de format :

- doit être placé entre apostrophes et différencie les majuscules et minuscules,
- peut inclure tout élément valide de format de date,
- comporte un élément *fm* qui supprime les espaces de remplissage ou les zéros de tête,
- est séparé de la valeur de date par une virgule.

ORACLE

3-31

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Afficher une date dans un format spécifique

Auparavant, toutes les dates Oracle s'affichaient au format DD-MON-YY. La fonction TO_CHAR vous permet de convertir ces dates en un autre format de votre choix.

Règles

- Le modèle de format doit être placé entre apostrophes et différencie les majuscules et minuscules.
- Il peut comprendre tout élément valide de format de date. N'oubliez pas de séparer la date du modèle de format par une virgule.
- Les noms de jours et de mois sont automatiquement complétés par des espaces.
- Pour supprimer les espaces de remplissage ou les zéros de tête, utilisez l'élément *fm* du mode de remplissage.
- La commande iSQL*Plus COLUMN (présentée dans un chapitre ultérieur) permet de mettre en forme les champs de caractères obtenus.

```
SELECT employee_id, TO_CHAR(hire_date, 'MM/YY') Month_Hired  
FROM   employees  
WHERE  last_name = 'Higgins';
```

EMPLOYEE_ID	MONTH
205	06/94

Eléments du modèle de format de date

YYYY	4 chiffres de l'année
YEAR	Année exprimée en toutes lettres
MM	Mois représenté par 2 chiffres
MONTH	Mois exprimé en toutes lettres
MON	Mois abrégé en trois lettres
DY	Jour abrégé en trois lettres
DAY	Jour exprimé en toutes lettres
DD	Valeur numérique correspondant au jour du mois

ORACLE®

Exemple d'éléments valides de format de date

Elément	Description
SCC ou CC	Siècle (le serveur fait précéder les dates av. J-C. du signe -)
Années au format YYYY ou SYYYY	Année (le serveur fait précéder les dates av. J-C. du signe -)
YYY, YY ou Y	Derniers chiffres de l'année
Y,YYY	Année avec virgule à cette position
IYYY, IYY, IY, I	Année en un à quatre chiffres selon la norme ISO
SYEAR ou YEAR	Année exprimée en toutes lettres (le serveur fait précéder les dates av. J-C. du signe -)
BC ou AD	Indicateur av. J.-C./apr. J.-C
B.C. ou A.D.	Indicateur av. J.-C./apr. J.-C. exprimé avec des points
Q	Trimestre
MM	Mois représenté par deux chiffres
MONTH	Nom du mois complété par des espaces afin d'atteindre une longueur de neuf caractères
MON	Mois abrégé en trois lettres
RM	Mois en chiffres romains
WW ou W	Semaine de l'année ou du mois
DDD, DD ou D	Jour de l'année, du mois ou de la semaine
DAY	Nom du jour complété par des espaces afin d'atteindre une longueur de neuf caractères
DY	Jour abrégé en trois lettres
J	Jour julien (nombre de jours depuis le 31 décembre 4713 av. J.-C.)

Eléments du modèle de format de date

- Les éléments horaires permettent de formater la partie horaire de la date.

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

- Vous pouvez ajouter des chaînes de caractères placées entre guillemets.

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

- Les suffixes numériques permettent d'indiquer les nombres en toutes lettres.

ddspth	fourteenth
--------	------------

ORACLE

Eléments de format de date – Formats horaires

Utilisez les formats présentés dans les tableaux ci-dessous pour afficher les informations et les littéraux relatifs à l'heure et pour changer les numéraux en nombres en toutes lettres.

Elément	Description
AM ou PM	Indicateur méridien
A.M. ou P.M.	Indicateur méridien exprimé avec des points
HH, HH12 ou HH24	Heure du jour ou heure suivant un format de 12 heures (1-12) ou de 24 heures (0-23)
MI	Minute (0-59)
SS	Seconde (0-59)
SSSS	Secondes après minuit (0-86399)

Autres formats

Elément	Description
/ . ,	La ponctuation est reproduite dans le résultat
“of the”	La chaîne placée entre guillemets est reproduite dans le résultat

Indiquer des suffixes s'appliquant à l'affichage des nombres

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

Utiliser la fonction TO_CHAR avec les dates

```
SELECT last_name,  
       TO_CHAR(hire_date, 'fmDD Month YYYY')  
          AS HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	17 June 1987
Kochhar	21 September 1989
De Haan	13 January 1993
Hunold	3 January 1990
Ernst	21 May 1991
Lorentz	7 February 1999
Mourgos	16 November 1999
...	

20 rows selected.

ORACLE

3-36

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction TO_CHAR avec les dates

L'instruction SQL de la diapositive affiche le nom et la date d'embauche de tous les employés. La date d'embauche s'affiche au format suivant : 17 juin 1987.

Exemple

Modifiez l'exemple de la diapositive afin d'afficher les dates au format suivant : Seventh of June 1994 12:00:00 AM.

```
SELECT last_name,  
       TO_CHAR(hire_date,  
              'fmDdspth "of" Month YYYY fmHH:MI:SS AM')  
          AS HIREDATE  
FROM   employees;
```

LAST_NAME	HIREDATE
King	Seventeenth of June 1987 12:00:00 AM
Kochhar	Twenty-First of September 1989 12:00:00 AM
...	
Higgins	Seventh of June 1994 12:00:00 AM
Gietz	Seventh of June 1994 12:00:00 AM

20 rows selected.

Vous remarquerez que le mois s'affiche selon le modèle de format précisé : la première lettre est en majuscule et les autres en minuscules.

Utiliser la fonction TO_CHAR avec les nombres

`TO_CHAR(number, 'format_model')`

Vous pouvez utiliser certains éléments de format avec la fonction TO_CHAR pour afficher une valeur numérique sous forme de caractère.

9	Représente un nombre
0	Force l'affichage d'un zéro
\$	Place un signe dollar flottant
L	Utilise le symbole monétaire local flottant
.	Affiche un séparateur décimal
,	Affiche un indicateur de milliers

ORACLE

3-37

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction TO_CHAR avec les nombres

Lorsque vous utilisez des valeurs numériques telles que des chaînes de caractères, vous devez leur affecter le type de données alphanumérique à l'aide de la fonction **TO_CHAR**, qui convertit une valeur de **type de données NUMBER** en VARCHAR2. Cette technique s'avère particulièrement utile dans le cas de la concaténation.

Eléments de format numériques

Si vous convertissez un nombre en donnée alphanumérique, vous pouvez utiliser les éléments de format suivants :

Elément	Description	Exemple	Résultat
9	Position numérique (le nombre de 9 détermine la largeur de l'affichage)	999999	1234
0	Affichage des zéros de tête	099999	001234
\$	Signe dollar flottant	\$999999	\$1234
L	Symbole monétaire local flottant	L999999	FF1234
.	Séparateur décimal à la position indiquée	999999.99	1234.00
,	Virgule à la position indiquée	999,999	1,234
MI	Signe moins à droite (valeurs négatives)	999999MI	1234-
PR	Mise entre parenthèses des nombres négatifs	999999PR	<1234>
EEEE	Notation scientifique (le format doit inclure quatre E)	99.999EEEE	1.234E+03
V	Multiplication par 10 <i>n</i> fois (<i>n</i> correspond au nombre de 9 après V)	9999V99	123400
B	Affichage des valeurs nulles sous forme d'espaces et non de 0	B9999.99	1234.00

Utiliser la fonction TO_CHAR avec les nombres

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY  
FROM employees  
WHERE last_name = 'Ernst';
```

SALARY
\$6,000.00

ORACLE

Règles

- Le serveur Oracle affiche une chaîne de **dîses** (#) à la place d'un nombre dont la quantité de chiffres dépasse celle indiquée dans le modèle de format.
- Le serveur Oracle arrondit la valeur décimale stockée au nombre d'espaces décimaux indiqué dans le modèle de format.

Utiliser les fonctions TO_NUMBER et TO_DATE

- Convertissez une chaîne de caractères en un format numérique à l'aide de la fonction TO_NUMBER :

```
TO_NUMBER(char[, 'format_model'])
```

- Convertissez une chaîne de caractères en un format de date à l'aide de la fonction TO_DATE :

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

- Ces fonctions possèdent un modificateur *fx* qui indique la correspondance exacte de l'argument alphanumérique et du modèle de format de date d'une fonction TO_DATE.

ORACLE

3-39

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions TO_NUMBER et TO_DATE

Vous souhaiterez peut-être convertir une chaîne de caractères en nombre ou en date. Pour cela, utilisez la fonction TO_NUMBER ou TO_DATE. Le modèle de format que vous choisissez est basé sur les éléments de format présentés précédemment.

Le modificateur "fx" indique la correspondance exacte de l'argument alphanumérique et du modèle de format de date d'une fonction TO_DATE.

- La ponctuation et le texte entre apostrophes de l'argument alphanumérique doivent être totalement identiques (excepté en ce qui concerne la casse) à ceux des parties correspondantes du modèle de format.
- L'argument alphanumérique ne peut pas comporter d'espaces supplémentaires. Oracle ignore ces derniers si *fx* n'est pas précisé.
- Les données numériques de l'argument alphanumérique doivent comporter le même nombre de chiffres que l'élément correspondant du modèle de format. Si *fx* n'est pas précisé, les nombres de l'argument alphanumérique peuvent omettre les zéros de tête.

Fonctions TO_NUMBER et TO_DATE (suite)

Exemple

Affichez le nom et la date d'embauche de tous les employés recrutés le 24 mai 1999. Puisque le modificateur `fx` est utilisé, une correspondance exacte est requise et les espaces situés après le mot "May" sont ignorés.

```
SELECT last_name, hire_date
  FROM employees
 WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY');

 WHERE hire_date = TO_DATE('May 24, 1999', 'fxMonth DD, YYYY')
 *
ERROR at line 3:
ORA-01858: a non-numeric character was found where a numeric was expected
```

Format de date RR

Année en cours	Date indiquée	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 à deux chiffres indiquée est la suivante :	
		0–49	50–99
Si l'année en cours comporte deux chiffres parmi les suivants :	0–49	La date renvoyée appartient au siècle en cours	La date renvoyée appartient au siècle qui précède le siècle en cours
	50–99	La date renvoyée appartient au siècle qui suit le siècle en cours	La date renvoyée appartient au siècle en cours

ORACLE®

Elément de format de date RR

Le **format de date RR** est similaire à l'élément YY, mais il permet d'indiquer différents siècles. Vous pouvez utiliser l'élément de format de date RR au lieu de YY pour faire varier le siècle de la valeur renvoyée en fonction de l'année à deux chiffres précisée et des deux derniers chiffres de l'année en cours. Le tableau de la diapositive synthétise le comportement de l'élément RR.

Année en cours	Date indiquée	Interprétation (RR)	Interprétation (YY)
1994	27-OCT-95	1995	1995
1994	27-OCT-17	2017	1917
2001	27-OCT-17	2017	2017

Exemple de format de date RR

Pour rechercher des employés embauchés avant 1990, utilisez le format RR, qui produit les mêmes résultats que la commande soit exécutée en 1999 ou à présent :

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM employees
WHERE hire_date < TO_DATE('01-Jan-90', 'DD-Mon-RR');
```

LAST_NAME	TO_CHAR(HIR)
King	17-Jun-1987
Kochhar	21-Sep-1989
Whalen	17-Sep-1987

ORACLE

Exemple d'élément de format de date RR

Vous pouvez utiliser le **format RR** pour rechercher des employés embauchés avant 1990. Puisque l'année est désormais postérieure à 1999, le format RR considère que la partie année de la date est comprise entre 1950 et 1999.

En revanche, la commande suivante ne sélectionne aucune ligne, car le format YY considère que la partie année de la date appartient au siècle en cours (2090).

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-yyyy')
FROM   employees
WHERE  TO_DATE(hire_date, 'DD-Mon-yy') < '01-Jan-1990';

no rows selected
```

Imbriquer des fonctions

- Les fonctions monoligne peuvent être imbriquées à tous les niveaux.
- Les fonctions imbriquées sont évaluées de l'intérieur vers l'extérieur.

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



ORACLE

3-43

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Imbriquer des fonctions

Les fonctions monoligne peuvent être imbriquées à tous les niveaux. Les **fonctions imbriquées** sont évaluées de l'intérieur vers l'extérieur. Des exemples présentés par la suite illustrent la flexibilité de ces fonctions

Imbriquer des fonctions

```
SELECT last_name,  
       NVL(TO_CHAR(manager_id), 'No Manager')  
  FROM employees  
 WHERE manager_id IS NULL;
```

LAST_NAME	NVL(TO_CHAR(MANAGER_ID),'NOMANAGER')
King	No Manager

ORACLE

3-44

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Imbriquer des fonctions (suite)

L'exemple de la diapositive permet d'afficher le directeur de la société, qui n'a pas de manager.
L'évaluation de l'instruction SQL s'effectue en deux étapes :

1. Evaluez la fonction intérieure pour convertir une valeur numérique en chaîne de caractères.
 - Result1 = TO_CHAR(manager_id)
2. Evaluez la fonction extérieure pour remplacer la valeur NULL par une chaîne de caractères.
 - NVL(Result1, 'No Manager')

L'ensemble de l'expression constitue l'en-tête de colonne, car aucun alias de colonne n'a été indiqué.

Exemple

Affichez la date du prochain vendredi qui se trouve six mois après la date d'embauche. La date obtenue doit s'afficher au format suivant : Friday, August 13th, 1999. Classez les résultats par date d'embauche.

```
SELECT    TO_CHAR(NEXT_DAY(ADD_MONTHS  
                           (hire_date, 6), 'FRIDAY'),  
                           'fmDay, Month DDth, YYYY')  
                           "Next 6 Month Review"  
  FROM      employees  
 ORDER BY  hire_date;
```

Fonctions générales

Ces fonctions, qui se rapportent à l'utilisation de valeurs NULL, s'utilisent avec tous les types de données.

- **NVL (expr1, expr2)**
- **NVL2 (expr1, expr2, expr3)**
- **NULLIF (expr1, expr2)**
- **COALESCE (expr1, expr2, ..., exprn)**

ORACLE

3-45

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions générales

Ces fonctions, qui se rapportent à l'utilisation de valeurs NULL dans la liste d'expressions, s'utilisent avec tous les types de données.

Fonction	Description
NVL	Convertit une valeur NULL en valeur réelle.
NVL2	Si expr1 n'est pas NULL, NVL2 renvoie expr2. Si expr1 est NULL, NVL2 renvoie expr3. L'argument expr1 accepte tous les types de données.
NULLIF	Compare deux expressions et renvoie NULL si elles sont égales ou la première si elles sont différentes.
COALESCE	Renvoie la première expression non NULL de la liste d'expressions.

Remarque : Pour plus d'informations sur les nombreuses fonctions disponibles, voir *Oracle9i SQL Reference*, "Functions".

Fonction NVL

Cette fonction convertit une valeur NULL en valeur réelle.

- Les types de données admis sont DATE, CHARACTER et NUMBER. .
- Les types de données doivent correspondre :
 - NVL(*commission_pct*,0)
 - NVL(*hire_date*, '01-JAN-97')
 - NVL(*job_id*, 'No Job Yet')

ORACLE®

3-46

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction NVL

Utilisez la **fonction NVL** pour convertir une valeur NULL en valeur réelle.

Syntaxe

NVL (*expr1*, *expr2*)

Explication de la syntaxe :

expr1 désigne la valeur ou l'expression source pouvant contenir une valeur NULL
expr2 désigne la valeur cible pour la conversion de la valeur NULL

La fonction NVL permet de convertir tous les types de données, mais le type de données de la valeur renvoyée est toujours le même que celui de *expr1*.

Conversions NVL de différents types de données

Type de données	Exemple de conversion
NUMBER	NVL(<i>number_column</i> , 9)
DATE	NVL(<i>date_column</i> , '01-JAN-95')
CHAR or VARCHAR2	NVL(<i>character_column</i> , 'Unavailable')

Utiliser la fonction NVL

```
SELECT last_name, salary, NVL(commission_pct, 0),  
       (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL  
FROM employees;
```

LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
King	24000	0	288000
Kochhar	17000	0	204000
De Haan	17000	0	204000
Hunold	9000	0	108000
Ernst	6000	0	72000
Lorentz	4200	0	50400
Mourgos	5800	0	69600
Rajs	3500	0	42000
...			

20 rows selected.



ORACLE

3-47

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction NVL (suite)

Pour calculer le salaire annuel de tous les employés, vous devez multiplier le salaire mensuel par 12, puis ajouter le pourcentage de commission.

```
SELECT last_name, salary, commission_pct,  
       (salary*12) + (salary*12*commission_pct) AN_SAL  
FROM   employees;
```

LAST_NAME	SALARY	COMMISSION_PCT	AN_SAL
Vargas	2500		
Zlotkey	10500	.2	151200
Abel	11000	.3	171600
Taylor	8600	.2	123840
Gietz	8300		

20 rows selected.

Vous remarquerez que le salaire annuel est calculé uniquement pour les employés qui touchent une commission. Si la valeur d'une colonne d'une expression est NULL, le résultat est NULL. Pour calculer les valeurs pour tous les employés, vous devez convertir la valeur NULL en nombre avant d'appliquer l'opérateur arithmétique. Dans l'exemple de la diapositive, la fonction NVL permet de convertir les

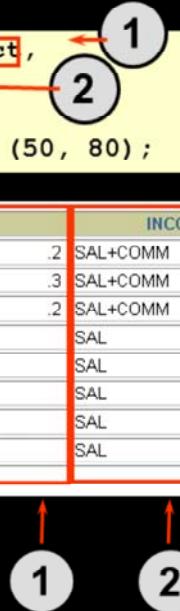
valeurs NULL en zéros.

Utiliser la fonction NVL2

```
SELECT last_name, salary, commission_pct,  
      NVL2(commission_pct,  
            'SAL+COMM', 'SAL') income  
FROM   employees WHERE department_id IN (50, 80);
```

LAST_NAME	SALARY	COMMISSION_PCT	INCOME
Zlotkey	10500		.2 SAL+COMM
Abel	11000		.3 SAL+COMM
Taylor	8600		.2 SAL+COMM
Mourgos	5800		SAL
Rajs	3500		SAL
Davies	3100		SAL
Matos	2600		SAL
Vargas	2500		SAL

8 rows selected.



ORACLE

Fonction NVL2

La **fonction NVL2** analyse la première expression. Si celle-ci n'est pas NULL, la fonction renvoie la deuxième expression. Si la première expression est NULL, la fonction renvoie la troisième expression.

Syntaxe

```
NVL(expr1, expr2, expr3)
```

In the syntax:

- | | |
|-------|---|
| expr1 | désigne la valeur ou l'expression source pouvant contenir une valeur NULL |
| expr2 | désigne la valeur renvoyée si expr1 n'est pas NULL |
| expr3 | désigne la valeur renvoyée si expr1 est NULL |

Dans l'exemple ci-dessus, la colonne COMMISSION_PCT est analysée. Si une valeur est détectée, la deuxième expression de SAL+COMM est renvoyée. Si la colonne COMMISSION_PCT contient une valeur NULL, la troisième expression de SAL est renvoyée.

L'argument *expr1* accepte tous les types de données. Les arguments *expr2* et *expr3* acceptent tous les types de données excepté LONG. Si les types de données d'*expr2* et *expr3* sont différents, le serveur Oracle convertit le type de données d'*expr3* en celui d'*expr2* avant de les comparer, à moins qu'*expr3* ne soit une constante de type NULL. Dans ce cas, il n'est pas nécessaire de convertir le type de données.

Le type de données de la valeur renvoyée est toujours identique à celui d'*expr2*, à moins qu'*expr2* ne

soit de type alphanumérique. Dans ce cas, la valeur renvoyée est de type VARCHAR2.

Utiliser la fonction NULLIF

```
1  
SELECT first_name, LENGTH(first_name) "expr1",  
      last_name, LENGTH(last_name) "expr2",  
      NULLIF(LENGTH(first_name), LENGTH(last_name)) result  
FROM   employees;
```

2
3

FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
Steven	6	King	4	6
Neena	5	Kochhar	7	5
Lex	3	De Haan	7	3
Alexander	9	Hunold	6	9
Bruce	5	Ernst	5	
Diana	5	Lorentz	7	5
Kevin	5	Mourgos	7	5
Trenna	6	Rajs	4	6
Curtis	6	Davies	6	
...				

20 rows selected.

3-49

Copyright © Oracle Corporation, 2001. Tous droits réservés.

ORACLE

Fonction NULLIF

La fonction **NULLIF** compare deux expressions. Si les expressions correspondent, la fonction renvoie NULL. Sinon, elle renvoie la première expression. Vous ne pouvez pas indiquer le littéral NULL pour la première expression.

Syntaxe

```
NULLIF (expr1, expr2)
```

Explication de la syntaxe :

expr1 désigne la valeur source comparée à *expr2*

expr2 désigne la valeur source comparée à *expr1* (si les expressions ne correspondent pas, *expr1* est renvoyée)

Dans l'exemple ci-dessus, l'ID de poste de la table EMPLOYEES est comparé à celui de la table JOB_HISTORY pour tous les employés qui figurent dans les deux tables. Le résultat affiche le poste actuel de l'employé. Si l'employé est répertorié plusieurs fois, cela signifie qu'il a occupé au moins deux postes auparavant.

Remarque : La fonction NULLIF est logiquement équivalente à l'expression CASE suivante. L'expression CASE est présentée plus loin dans ce chapitre.

```
CASE WHEN expr1 = expr 2 THEN NULL ELSE expr1 END
```

Utiliser la fonction COALESCE

- L'avantage de la fonction COALESCE par rapport à la fonction NVL réside dans le fait qu'elle peut accepter plusieurs valeurs différentes.
- La fonction renvoie la première expression si celle-ci n'est pas NULL. Sinon, elle fusionne les expressions restantes.

ORACLE

3-50

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction COALESCE

La fonction COALESCE renvoie la première expression non NULL de la liste.

Syntaxe

COALESCE (*expr1, expr2, ... exprn*)

Explication de la syntaxe :

expr1 renvoie cette expression si elle n'est pas NULL

expr2 renvoie cette expression si elle n'est pas NULL et si la première expression est NULL

exprn renvoie cette expression si les expressions précédentes sont NULL

Utiliser la fonction COALESCE

```
SELECT    last_name,  
          COALESCE(commission_pct, salary, 10) comm  
FROM      employees  
ORDER BY commission_pct;
```

LAST_NAME	COMM
Grant	.15
Zlotkey	.2
Taylor	.2
Abel	.3
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
...	

20 rows selected.

ORACLE

3-51

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction COALESCE (suite)

Dans l'exemple ci-dessus, la valeur COMMISSION_PCT s'affiche si elle n'est pas NULL. Si elle est NULL, SALARY s'affiche. Si COMMISSION_PCT et SALARY sont NULL, la valeur 10 s'affiche.

Expressions conditionnelles

- Ces expressions permettent d'utiliser la logique IF-THEN-ELSE dans une instruction SQL.
- Elles utilisent deux méthodes :
 - Expression CASE
 - Fonction DECODE

ORACLE

3-52

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Expressions conditionnelles

L'expression CASE et la fonction DECODE sont deux méthodes utilisées pour implémenter un traitement conditionnel (logique IF-THEN-ELSE) dans une instruction SQL.

Remarque : L'**expression CASE** est une nouvelle expression dans la version Oracle9*i* Server. Elle est conforme au langage SQL ANSI tandis que la fonction **DECODE** est spécifique à la syntaxe Oracle.

Expression CASE

Cette expression, qui fonctionne comme une instruction IF-THEN-ELSE, facilite les interrogations conditionnelles :

```
CASE expr WHEN comparison_expr1 THEN return_expr1
            [WHEN comparison_expr2 THEN return_expr2
             WHEN comparison_exprn THEN return_exprn
             ELSE else_expr]
END
```

ORACLE

3-53

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Expression CASE

Les **expressions CASE** vous permettent d'utiliser la logique IF-THEN-ELSE dans des instructions SQL sans appeler de procédures.

Dans une expression CASE simple, Oracle recherche la première instruction WHEN . . . THEN pour laquelle expr correspond à comparison_expr et renvoie return_expr. Si aucune instruction ne remplit cette condition et qu'il existe une clause ELSE, Oracle renvoie else_expr. Sinon, il renvoie NULL. Vous ne pouvez pas indiquer de littéral NULL pour return_expr et else_expr.

Toutes les expressions (expr, comparison_expr et return_expr) doivent présenter le même type de données (CHAR, VARCHAR2, NCHAR ou NVARCHAR2).

Utiliser l'expression CASE

Cette expression, qui fonctionne comme une instruction IF-THEN-ELSE, facilite les interrogations conditionnelles :

```
SELECT last_name, job_id, salary,  
      CASE job_id WHEN 'IT_PROG' THEN 1.10*salary  
                    WHEN 'ST_CLERK' THEN 1.15*salary  
                    WHEN 'SA_REP'   THEN 1.20*salary  
                    ELSE          salary END    "REVISED_SALARY"  
FROM employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
Gietz	AC_ACCOUNT	8300	8300

Utiliser l'expression CASE

L'instruction SQL précédente décode la valeur de JOB_ID. L'augmentation de salaire est de 10 % si cette valeur correspond à IT_PROG, de 15 % si elle correspond à ST_CLERK et de 20 % si elle correspond à SA_REP. Aucune augmentation de salaire n'est appliquée aux autres fonctions.

La même instruction peut être écrite à l'aide de la fonction DECODE.

Fonction DECODE

Cette fonction, qui fonctionne comme une instruction CASE ou IF-THEN-ELSE, facilite les interrogations conditionnelles :

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

ORACLE

3-55

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonction DECODE

La **fonction DECODE** permet de décoder les expressions selon une méthode similaire à celle de la logique IF-THEN-ELSE utilisée dans de nombreux langages. Elle décode l'expression (*expression*) après l'avoir comparée à chaque valeur de recherche (*search*). Si l'expression est identique à *search*, le résultat (*result*) est renvoyé.

En l'absence de valeur par défaut, une valeur NULL est renvoyée à chaque fois qu'une valeur de recherche ne correspond à aucune valeur de résultat.

Utiliser la fonction DECODE

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG', 1.10*salary,
              'ST_CLERK', 1.15*salary,
              'SA REP', 1.20*salary,
              salary)
  REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
Gietz	AC_ACCOUNT	8300	8300

Utiliser la fonction DECODE

L'instruction SQL précédente teste la valeur de JOB_ID. L'augmentation de salaire est de 10 % si cette valeur correspond à IT_PROG, de 15 % si elle correspond à ST_CLERK et de 20 % si elle correspond à SA REP. Aucune augmentation de salaire n'est appliquée aux autres fonctions.

Cette instruction peut être exprimée en pseudo-code sous la forme d'une instruction IF-THEN-ELSE :

```
IF job_id = 'IT_PROG'      THEN salary = salary*1.10
IF job_id = 'ST_CLERK'     THEN salary = salary*1.15
IF job_id = 'SA REP'       THEN salary = salary*1.20
ELSE salary = salary
```

Utiliser la fonction DECODE

Affichez le taux d'imposition applicable à chaque employé du service 80.

```
SELECT last_name, salary,  
       DECODE (TRUNC(salary/2000, 0),  
                0, 0.00,  
                1, 0.09,  
                2, 0.20,  
                3, 0.30,  
                4, 0.40,  
                5, 0.42,  
                6, 0.44,  
                0.45) TAX_RATE  
FROM   employees  
WHERE  department_id = 80;
```

ORACLE

3-57

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Exemple

La diapositive présente un autre exemple d'utilisation de la fonction DECODE. Le taux d'imposition de chaque employé du service 80 est déterminé en fonction du salaire mensuel. Les taux d'imposition sont conformes aux valeurs mentionnées dans les données suivantes.

Fourchette de salaires mensuels	Taux
\$0.00 - 1999.99	00%
\$2,000.00 - 3,999.99	09%
\$4,000.00 - 5,999.99	20%
\$6,000.00 - 7,999.99	30%
\$8,000.00 - 9,999.99	40%
\$10,000.00 - 11,999.99	42%
\$12,200.00 - 13,999.99	44%
\$14,000.00 or greater	45%

LAST_NAME	SALARY	TAX_RATE
Zlotkey	10500	.42
Abel	11000	.42
Taylor	8600	.4

Synthèse

Ce chapitre vous à permis d'apprendre à :

- effectuer des calculs sur des données à l'aide de fonctions
- modifier des éléments de données individuels à l'aide de fonctions
- manipuler la sortie de groupes de lignes à l'aide de fonctions
- modifier des formats de date pour l'affichage à l'aide de fonctions
- convertir les types de données de colonnes à l'aide de fonctions
- utiliser des fonctions NVL
- utiliser la logique IF-THEN-ELSE

ORACLE

3-58

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Fonctions monoligne

Les fonctions monoligne peuvent être imbriquées à tous les niveaux. Elles peuvent manipuler les éléments suivants :

- Données alphanumériques : LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH
- Données numériques : ROUND, TRUNC, MOD
- Données de type date : MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC
- Les valeurs de date peuvent également utiliser des opérateurs arithmétiques
- Les fonctions de conversion agissent sur les valeurs de date, alphanumériques et numériques : TO_CHAR, TO_DATE, TO_NUMBER
- Plusieurs fonctions se rapportent aux valeurs NULL, par exemple, NVL, NVL2, NULLIF et COALESCE
- La logique IF-THEN-ELSE peut être appliquée dans une instruction SQL à l'aide de l'expression CASE ou de la fonction DECODE.

SYSDATE et DUAL

La fonction de date SYSDATE renvoie la date et l'heure actuelles. Elle est généralement sélectionnée dans une table factice nommée DUAL.

Présentation de l'exercice 3, 2^{ème} partie

Dans cet exercice, vous allez :

- créer des interrogations utilisant des fonctions numériques, alphanumériques et de date
- utiliser la concaténation avec des fonctions
- écrire des interrogations qui ne différencient pas les majuscules et les minuscules pour tester l'utilité des fonctions alphanumériques
- effectuer des calculs relatifs aux années et aux mois d'ancienneté d'un employé
- déterminer la date de révision du salaire d'un employé

ORACLE

3-59

Copyright © Oracle Corporation, 2001. Tous droits réservés.

Exercice 3, 2^{ème} partie

La variété des exercices qui suivent va vous permettre d'appliquer les différentes fonctions utilisables avec des données de type alphanumérique, 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.

Exercice 3, 1^{ère} partie

1. Ecrivez une instruction pour afficher la date du jour. Nommez la colonne Date.

Date
28-SEP-01

2. Affichez pour chaque employé le numéro, le nom, le salaire et le salaire augmenté de 15 % sous forme de nombre. Nommez la colonne New Salary. Placez votre instruction SQL dans un fichier texte nommé lab3_2.sql.
 3. Exécutez l'instruction du fichier lab3_2.sql.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary
100	King	24000	27600
101	Kochhar	17000	19550
102	De Haan	17000	19550
103	Hunold	9000	10350
...			
202	Fay	6000	6900
205	Higgins	12000	13800
206	Gietz	8300	9545

20 rows selected.

4. Modifiez votre instruction lab3_2.sql pour ajouter une colonne permettant de soustraire l'ancien salaire du nouveau salaire. Nommez la colonne Increase. Enregistrez le contenu du fichier sous le nom lab3_4.sql, puis exécutez la nouvelle instruction.

EMPLOYEE_ID	LAST_NAME	SALARY	New Salary	Increase
100	King	24000	27600	3600
101	Kochhar	17000	19550	2550
102	De Haan	17000	19550	2550
103	Hunold	9000	10350	1350
104	Ernst	6000	6900	900
107	Lorentz	4200	4830	630
124	Mourgos	5800	6670	870
141	Rajs	3500	4025	525
142	Davies	3100	3565	465
143	Matos	2600	2990	390
...				
201	Hartstein	13000	14950	1950
202	Fay	6000	6900	900
205	Higgins	12000	13800	1800
206	Gietz	8300	9545	1245

20 rows selected.

Exercice 3, 1^{ère} partie (suite)

5. Ecrivez une instruction permettant d'afficher le nom des employés dont la première lettre est en majuscule et les suivantes en minuscules ainsi que la longueur des noms commençant par la lettre J, A ou M. Donnez à chaque colonne un intitulé approprié. Triez les résultats selon le nom de l'employé.

Name	Length
Abel	4
Matos	5
Mourgos	7

Exercice 3, 2^{ème} partie

6. Affichez le nom de chaque employé et calculez le nombre de mois écoulés entre la date du jour et 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.

Remarque : Vos résultats seront différents.

LAST_NAME	MONTHS_WORKED
Zlotkey	20
Mourgos	22
Grant	28
Lorentz	32
Vargas	39
Taylor	42
Matos	42
Fay	49
Davies	56
Abel	65
Hartstein	67
Rajs	71
Higgins	88
Gietz	88
LAST_NAME	MONTHS_WORKED
De Haan	105
Ernst	124
Hunold	141
Kochhar	144
Whalen	168
King	171

20 rows selected.

Exercice 3, 2^{ème} partie (suite)

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

Dream Salaries
King earns \$24,000.00 monthly but wants \$72,000.00.
Kochhar earns \$17,000.00 monthly but wants \$51,000.00.
De Haan earns \$17,000.00 monthly but wants \$51,000.00.
Hunold earns \$9,000.00 monthly but wants \$27,000.00.
Ernst earns \$6,000.00 monthly but wants \$18,000.00.
Lorentz earns \$4,200.00 monthly but wants \$12,600.00.
Mourgos earns \$5,800.00 monthly but wants \$17,400.00.
Rajs earns \$3,500.00 monthly but wants \$10,500.00.
Davies earns \$3,100.00 monthly but wants \$9,300.00.
Matos earns \$2,600.00 monthly but wants \$7,800.00.
Vargas earns \$2,500.00 monthly but wants \$7,500.00.
■ ■ ■
Gietz earns \$8,300.00 monthly but wants \$24,900.00.

20 rows selected.

S'il vous reste du temps, effectuez les exercices suivants :

8. Créez une instruction permettant d'afficher le nom et le salaire de tous les employés. Le salaire devra comporter 15 caractères et les caractères de remplissage seront insérés par la gauche sous forme de signes \$. Nommez la colonne SALARY.

LAST_NAME	SALARY
King	\$\$\$\$\$\$\$\$\$\$\$\$\$\$24000
Kochhar	\$\$\$\$\$\$\$\$\$\$\$\$\$\$17000
De Haan	\$\$\$\$\$\$\$\$\$\$\$\$\$\$17000
Hunold	\$\$\$\$\$\$\$\$\$\$\$\$\$\$9000
Ernst	\$\$\$\$\$\$\$\$\$\$\$\$\$\$6000
Lorentz	\$\$\$\$\$\$\$\$\$\$\$\$\$\$4200
Mourgos	\$\$\$\$\$\$\$\$\$\$\$\$\$\$5800
Rajs	\$\$\$\$\$\$\$\$\$\$\$\$\$\$3500
■ ■ ■	
Higgins	\$\$\$\$\$\$\$\$\$\$\$\$\$\$12000
Gietz	\$\$\$\$\$\$\$\$\$\$\$\$\$\$8300

20 rows selected.

Exercice 3, 2^{ème} partie (suite)

9. Affichez le nom et la date d'embauche de chaque employé ainsi que la date de révision du salaire, qui correspond au premier lundi après 6 mois d'activité. Nommez la colonne REVIEW. Les dates devront apparaître au format suivant : "Sunday, the Seventh of September, 2000"

LAST_NAME	HIRE_DATE	REVIEW
King	17-JUN-87	Monday, the Twenty-First of December, 1987
Kochhar	21-SEP-89	Monday, the Twenty-Sixth of March, 1990
De Haan	13-JAN-93	Monday, the Nineteenth of July, 1993
Hunold	03-JAN-90	Monday, the Ninth of July, 1990
Ernst	21-MAY-91	Monday, the Twenty-Fifth of November, 1991
Lorentz	07-FEB-99	Monday, the Ninth of August, 1999
Mourgos	16-NOV-99	Monday, the Twenty-Second of May, 2000
Rajs	17-OCT-95	Monday, the Twenty-Second of April, 1996
Davies	29-JAN-97	Monday, the Fourth of August, 1997
...		
Gietz	07-JUN-94	Monday, the Twelfth of December, 1994

20 rows selected.

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

LAST_NAME	HIRE_DATE	DAY
Grant	24-MAY-99	MONDAY
Ernst	21-MAY-91	TUESDAY
Mourgos	16-NOV-99	TUESDAY
Taylor	24-MAR-98	TUESDAY
Rajs	17-OCT-95	TUESDAY
Gietz	07-JUN-94	TUESDAY
Higgins	07-JUN-94	TUESDAY
King	17-JUN-87	WEDNESDAY
De Haan	13-JAN-93	WEDNESDAY
...		
Abel	11-MAY-96	SATURDAY
Lorentz	07-FEB-99	SUNDAY
Fay	17-AUG-97	SUNDAY
Matos	15-MAR-98	SUNDAY

20 rows selected.

Exercice 3, 2^{ème} partie (suite)

Pour aller plus loin, effectuez les exercices suivants :

11. Créez une instruction permettant d'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.

LAST_NAME	COMM
King	No Commission
Kochhar	No Commission
De Haan	No Commission
Hunold	No Commission
Ernst	No Commission
Lorentz	No Commission
Mourgos	No Commission
Rajs	No Commission
Davies	No Commission
Matos	No Commission
Vargas	No Commission
Zlotkey	.2
Abel	.3
Taylor	.2
...	
Gietz	No Commission

20 rows selected.

12. Créez une instruction permettant d'afficher le nom des employés et leur salaire annuel sous forme d'astérisques. Chaque astérisque représente mille dollars. Triez les données dans l'ordre décroissant des salaires. Nommez la colonne EMPLOYEE_AND_THEIR_SALARIES.

EMPLOYEE_AND_THEIR_SALARIES	
King *****	
Kochhar *****	
De Haan *****	
Hartstei *****	
Higgins *****	
Abel *****	
...	
Vargas **	

20 rows selected.

Exercice 3, 2^{ème} partie (suite)

13. A l'aide de la fonction DECODE, écrivez une instruction qui affiche l'échelon de tous les employés en fonction de la valeur de la colonne JOB_ID, conformément au données suivantes :

Fonction	Echelon
AD_PRES	A
ST_MAN	B
IT_PROG	C
SA REP	D
ST_CLERK	E
Aucun de ces postes	0

JOB_ID	G
AD_PRES	A
AD_VP	0
AD_VP	0
IT_PROG	C
IT_PROG	C
IT_PROG	C
ST_MAN	B
ST_CLERK	E
ST_CLERK	E
ST_CLERK	E
...	
AC_MGR	0
AC_ACCOUNT	0

20 rows selected.

14. Réécrivez l'instruction de la question précédente en utilisant la syntaxe CASE.