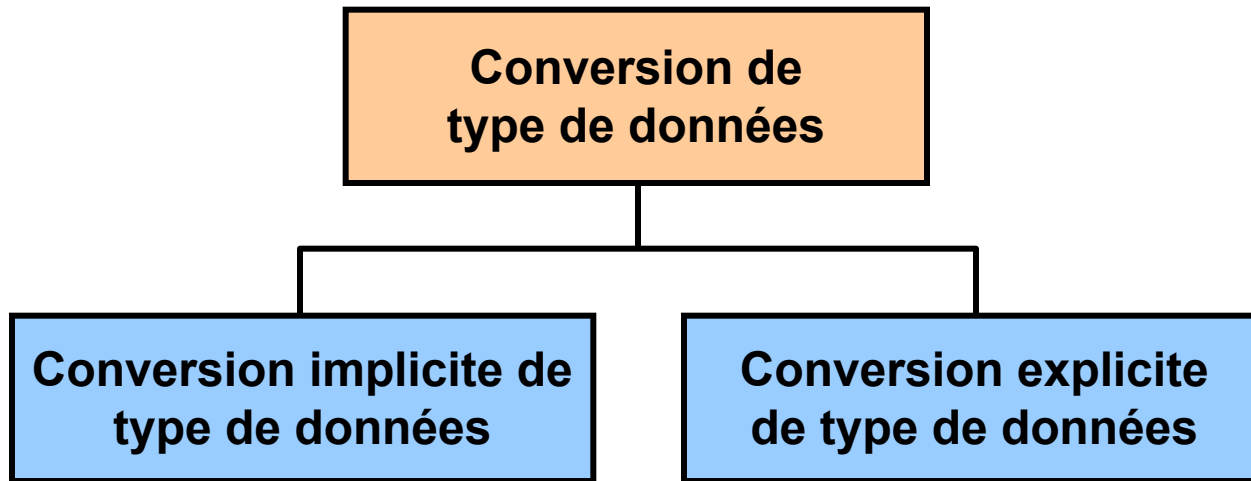


Utiliser des fonctions de
conversion
et des expressions conditionnelles

Fonctions de conversion



Conversion implicite de types de données

- Dans les expressions, le serveur Oracle peut convertir automatiquement les types de données ci-après :

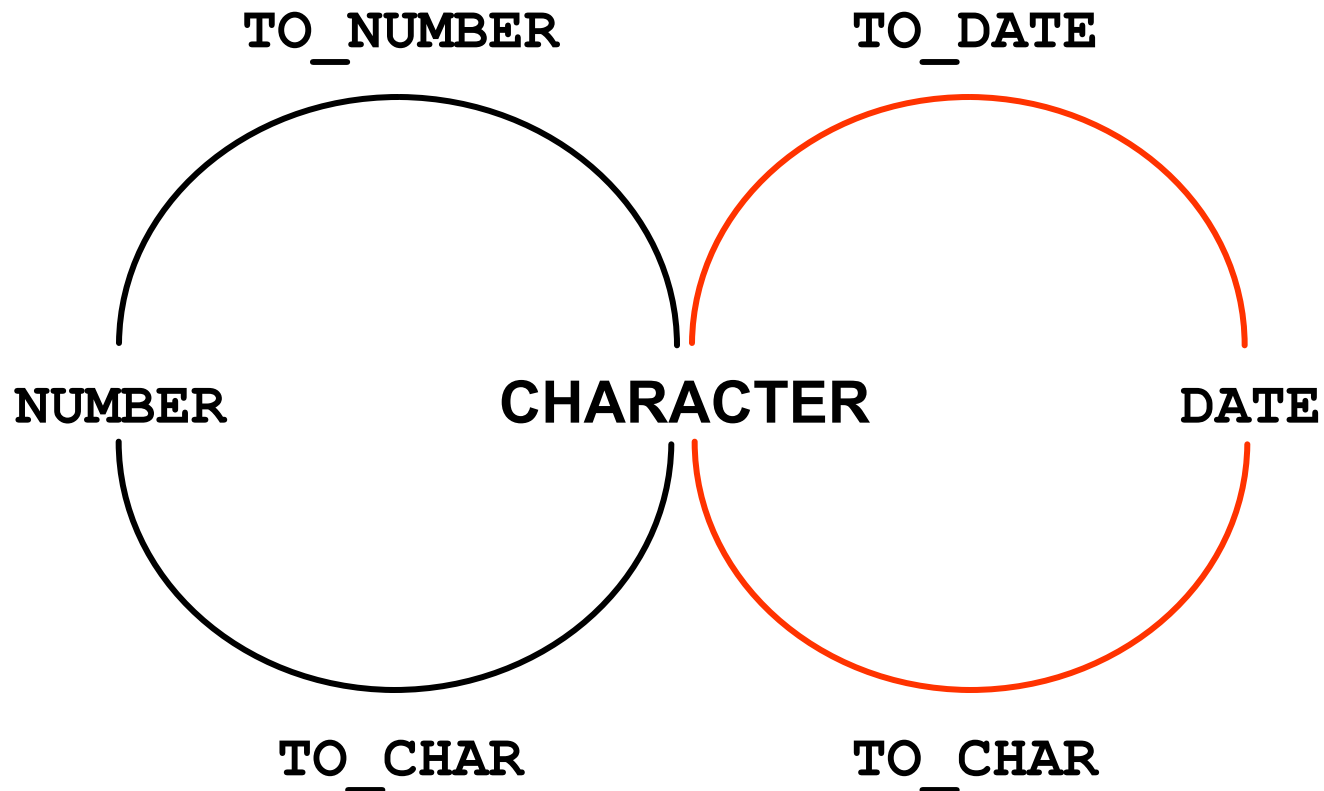
De	A
VARCHAR2 ou CHAR	NUMBER
VARCHAR2 ou CHAR	DATE

Conversion implicite de types de données

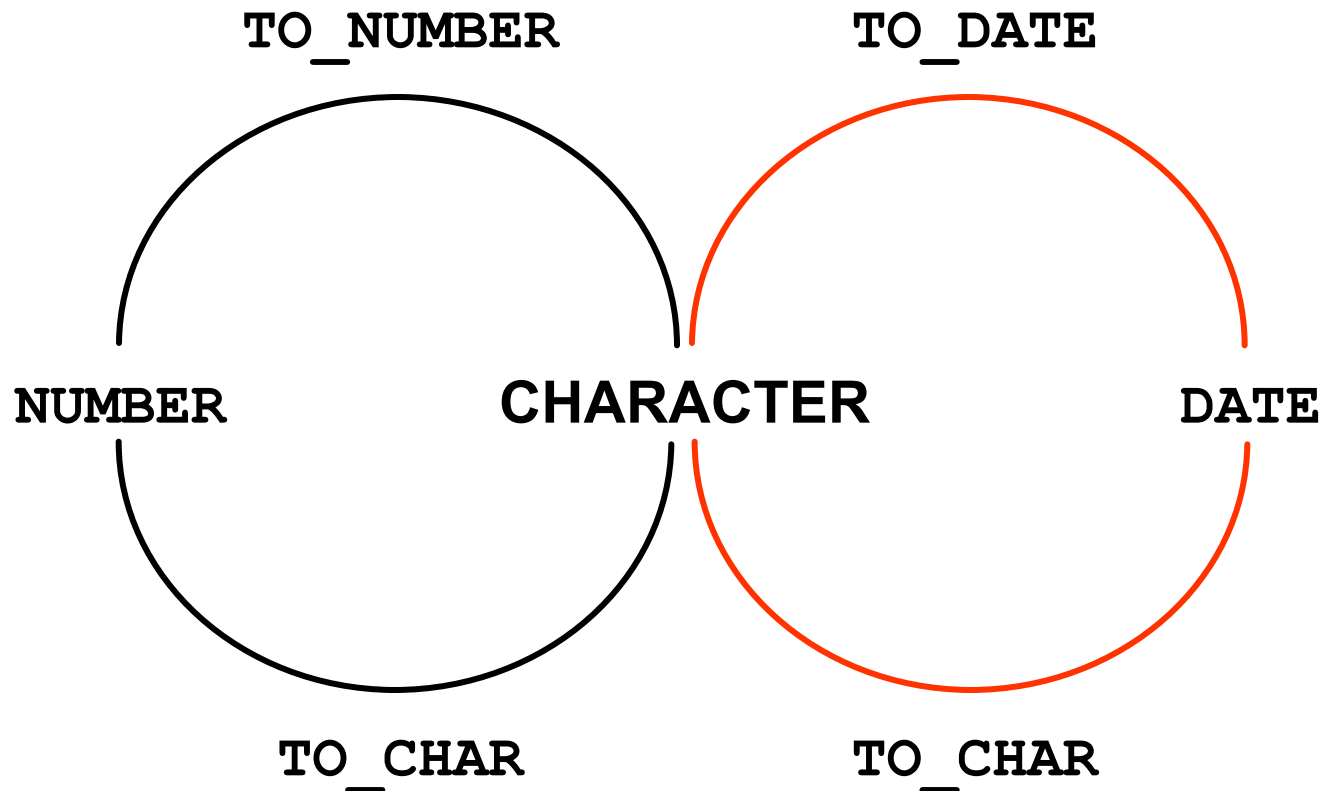
- Pour l'évaluation des expressions, le serveur Oracle peut convertir automatiquement les types de données suivants :

De	A
NUMBER	VARCHAR2 ou CHAR
DATE	VARCHAR2 ou CHAR

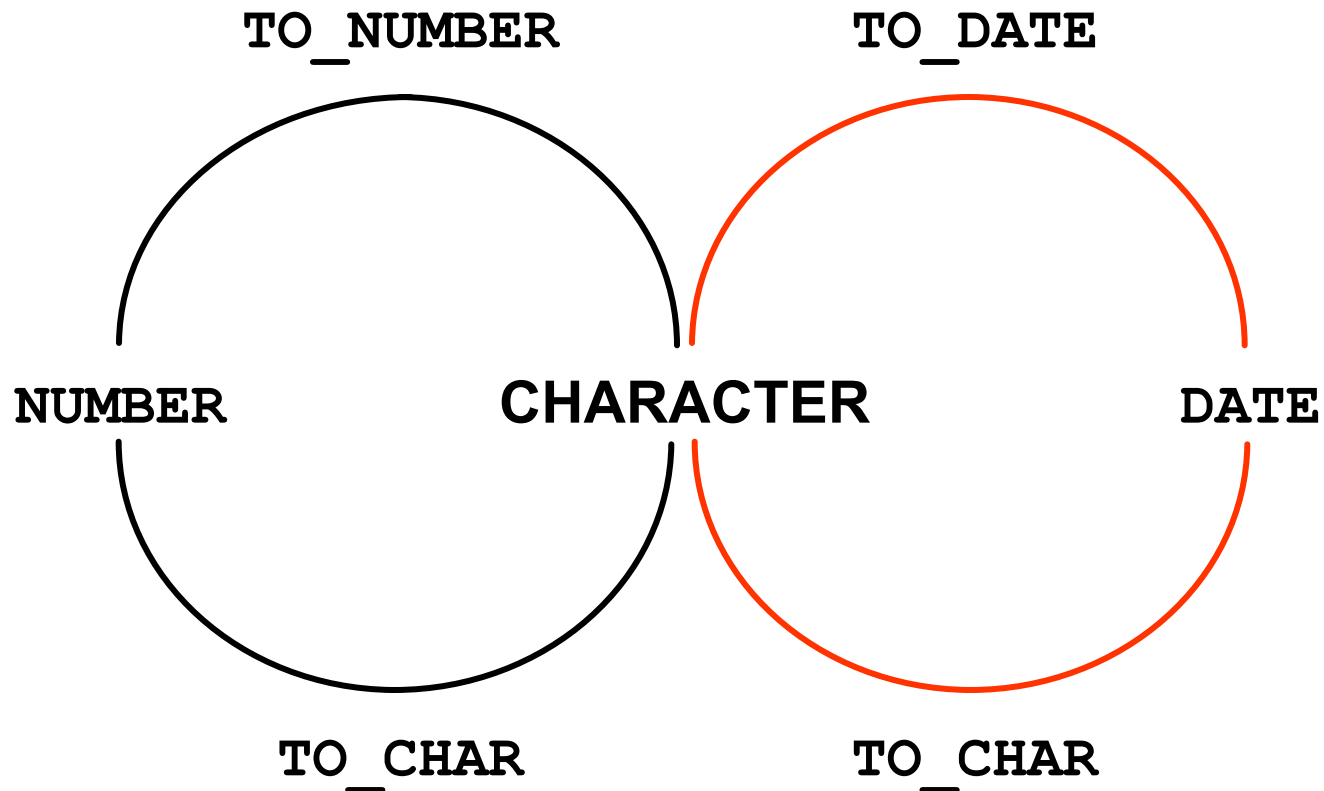
Conversion explicite de types de données



Conversion explicite de types de données



Conversion explicite de types de données



Utiliser la fonction TO_CHAR avec des dates

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

- Le modèle de format :
 - doit être indiqué entre apostrophes
 - distingue les majuscules des minuscules
 - peut inclure n'importe quel élément de format de date valide
 - comporte un élément `fm` permettant de supprimer les espaces de complément ou les zéros de début
 - est séparé de la valeur de date par une virgule

Éléments du modèle de format de date

Élément	Résultat
YYYY	Année complète en chiffres
YEAR	Année en toutes lettres (en anglais)
MM	Mois sur deux chiffres
MONTH	Mois en toutes lettres
MON	Abréviation à trois lettres du mois
DY	Abréviation à trois lettres du jour de la semaine
DAY	Jour de la semaine en toutes lettres
DD	Jour du mois sous forme numérique

Éléments du modèle de format de date

- Les éléments horaires formatent la partie heure de la date :

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

- Ajoutez des chaînes de caractères en les incluant entre guillemets :

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

- Des suffixes peuvent être ajoutés aux nombres en toutes lettres :

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

Utiliser la fonction TO_CHAR avec des dates

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

	LAST_NAME	HIREDATE
1	Whalen	17 September 1987
2	Hartstein	17 February 1996
3	Fay	17 August 1997
4	Higgins	7 June 1994
5	Gietz	7 June 1994
6	King	17 June 1987
7	Kochhar	21 September 1989
8	De Haan	13 January 1993
9	Hunold	3 January 1990
10	Ernst	21 May 1991

...

Utiliser la fonction TO_CHAR avec des nombres

```
TO_CHAR(number, 'format_model')  
--
```

- Voici quelques-uns des éléments de format que vous pouvez utiliser avec la fonction TO_CHAR pour afficher une valeur numérique sous la forme d'une chaîne de caractères :

Élément	Résultat
9	Représente un nombre
0	Force l'affichage d'un zéro
\$	Place un signe dollar flottant
L	Utilise le symbole flottant de la devise locale
.	Affiche un point comme séparateur décimal
,	Affiche une virgule comme séparateur des milliers

Utiliser la fonction TO_CHAR avec des nombres

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

	SALARY
1	\$6,000.00

Utiliser les fonctions TO_NUMBER et TO_DATE

- Convertir une chaîne de caractères en nombre à l'aide de la fonction TO_NUMBER :

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

- Convertir une chaîne de caractères en date à l'aide de la fonction TO_DATE :

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

- Ces fonctions comportent un modificateur `fx`. Celui-ci indique qu'il doit y avoir une correspondance exacte entre l'argument de type caractère et le modèle de format de date d'une fonction TO_DATE.

Utiliser les fonctions TO_CHAR et TO_DATE avec le format RR

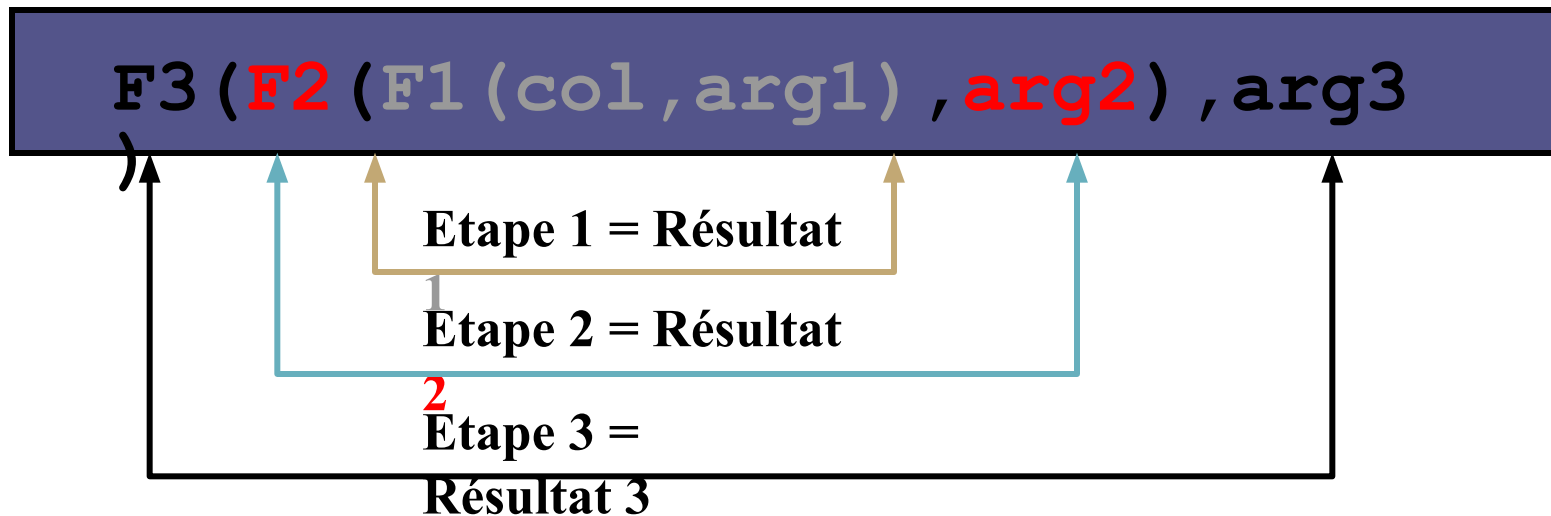
- Pour trouver les employés embauchés avant 1990, utilisez le format de date RR, qui produit les mêmes résultats que la commande soit exécutée en 1999 ou maintenant :

```
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(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

Fonctions d'imbrication

- Les fonctions monolignes peuvent être imbriquées à n'importe quel niveau.
- Les fonctions imbriquées sont évaluées du niveau le plus profond au niveau le moins profond.



Fonctions d'imbrication : Exemple 1

```
SELECT last_name,  
       UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))  
FROM   employees  
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

Fonctions d'imbrication : Exemple 2

```
SELECT      TO_CHAR(ROUND((salary/7), 2), '99G999D99',  
            'NLS_NUMERIC_CHARACTERS = ','.')  
            "Formatted Salary"  
FROM employees;
```

	Formatted Salary
1	628,57
2	1.857,14
3	857,14
4	1.714,29
5	1.185,71
6	3.428,57

...

Fonctions générales

- Les fonctions suivantes, qui peuvent être utilisées avec n'importe quel type de données, sont relatives à l'utilisation de valeurs NULL :
 - NVL (expr1, expr2)
 - NVL2 (expr1, expr2, expr3)
 - NULLIF (expr1, expr2)
 - COALESCE (expr1, expr2, ..., exprn)

Fonction NVL

- Convertit une valeur NULL en valeur réelle :
 - Il est possible d'utiliser des données de type nombre, caractère ou date.
 - Les types de données doivent correspondre :
 - `NVL(commission_pct, 0)`
 - `NVL(hire_date, '01-JAN-97')`
 - `NVL(job_id, 'No Job Yet')`

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
1	Whalen	4400	0	52800
2	Hartstein	13000	0	156000
3	Fay	6000	0	72000
4	Higgins	12000	0	144000
5	Gietz	8300	0	99600
6	King	24000	0	288000
7	Kochhar	17000	0	204000
8	De Haan	17000	0	204000
9	Hunold	9000	0	108000
10	Ernst	6000	0	72000

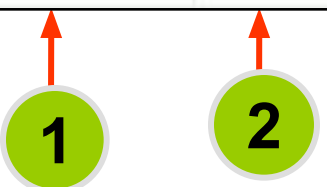
...

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);
```

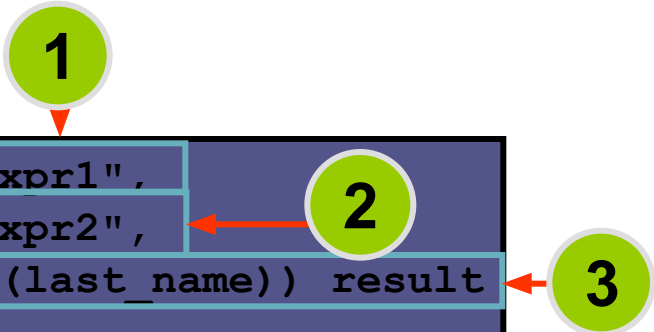


	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM



Utiliser la fonction NULLIF

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



	1 FIRST_NAME	2 expr1	3 LAST_NAME	4 expr2	5 RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
8	Michael	7	Hartstein	9	7
9	Shelley	7	Higgins	7	(null)

...

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 alternatives.
- Si la valeur de la première expression n'est pas NULL, la fonction COALESCE renvoie cette valeur. Sinon, la fonction examine successivement les valeurs des autres expressions jusqu'à ce qu'elle trouve une valeur non NULL.

Utiliser la fonction COALESCE

```
SELECT last_name, employee_id,  
       COALESCE(TO_CHAR(commission_pct), TO_CHAR(manager_id))  
       ,  
       'No commission and no manager')  
FROM employees;
```

	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COMMISSI...
1	Whalen	200	101
2	Hartstein	201	100
3	Fay	202	201
4	Higgins	205	101
5	Gietz	206	205
6	King	100	No commission and no manager

...

17	Zlotkey	149	.2
18	Abel	174	.3
19	Taylor	176	.2
20	Grant	178	.15

Expressions conditionnelles

- Elles permettent d'implémenter la logique `IF-THEN-ELSE` dans une instruction SQL.
- Deux méthodes possibles :
 - Expression `CASE`
 - Fonction `DECODE`

Expression CASE

- Elle facilite les interrogations conditionnelles en faisant le travail d'une instruction IF-THEN-ELSE :

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

Utiliser l'expression CASE

- Elle facilite les interrogations conditionnelles en faisant le travail d'une instruction IF-THEN-ELSE :

```
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
1	Whalen	AD_ASST	4400	4400
...				
9	Hunold	IT_PROG	9000	9900
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
14	Davies	ST_CLERK	3100	3565
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

Fonction DECODE

- Facilite les interrogations conditionnelles en faisant le travail d'une expression CASE ou d'une instruction IF-THEN-ELSE :

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

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
...				
10	Ernst	IT_PROG	6000	6600
11	Lorentz	IT_PROG	4200	4620
12	Mourgos	ST_MAN	5800	5800
13	Rajs	ST_CLERK	3500	4025
...				
19	Taylor	SA_REP	8600	10320
20	Grant	SA_REP	7000	8400

Utiliser la fonction DECODE

- Afficher le taux d'imposition applicable pour chaque employé du département 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;
```