Conversion Functions

Conversion functions convert a value from one datatype to another. Generally, the form of the function names follows the convention *datatype* TO *datatype*. The first datatype is the input datatype. The second datatype is the output datatype. The SQL conversion functions are:

ASCIISTR	RAWTONHEX	TO_NCHAR (character)
BIN_TO_NUM	ROWIDTOCHAR	TO_NCHAR (datetime)
CAST	ROWIDTONCHAR	TO_NCHAR (number)
CHARTOROWID	TO_CHAR (character)	TO_NCLOB
COMPOSE	TO_CHAR (datetime)	TO_NUMBER
CONVERT	TO_CHAR (number)	TO_SINGLE_BYTE
DECOMPOSE	TO_CLOB	TO_YMINTERVAL
HEXTORAW	TO_DATE	TRANSLATE ... USING
NUMTODSINTERVAL	TO_DSINTERVAL	UNISTR
NUMTOYMINTERVAL	TO_LOB	
RAWTOHEX	TO_MULTI_BYTE	

Conversion Functions

Datatype conversion

1. **Implicit datatype conversion**
2. **Explicit datatype conversion**

Conversion Functions

in addition to Oracle datatypes, columns of tables in an Oracle8 database can be defined usiıiü ANSI. DB2, and SQL/DS datattypes. However, ths Oracle Server internally converts such datatypes to OracIe8 datatypes.

In some cases, Oracle Server allows data of one datatype where it expects data of a different datatype. This is allowed when Oracle Server can automatically converts the data to the expected datatype. This datatype conversion can be done *ıimplicitly* by Oracle Server or *explicitly* by the user.

Implicit datatvpe conversions work according to the rules explained in next two slides.

Explicit datatype conversions are done by using the conversion functions. Conversion functions convert a value from one datatype to another. Generally, the form of the function names follows the convention datatype TO *datatype.* The first datatype is the input dataty; the last datatype is the output.

Implicit Datatype Conversion

For assignments, the Oracle can automatically convert the follovving:

From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE
NUMBER	VARCHAR2
DATE	VARCHAR2

Implicit Datatype Conversion

The assignment succeeds if the Oracle Server can convert the dalatype of the value used in the assignment to that of the assignment target.

Implicit Datatype Conversion

For expression evaluation, the Oracle Server can automatically convert the follovving:

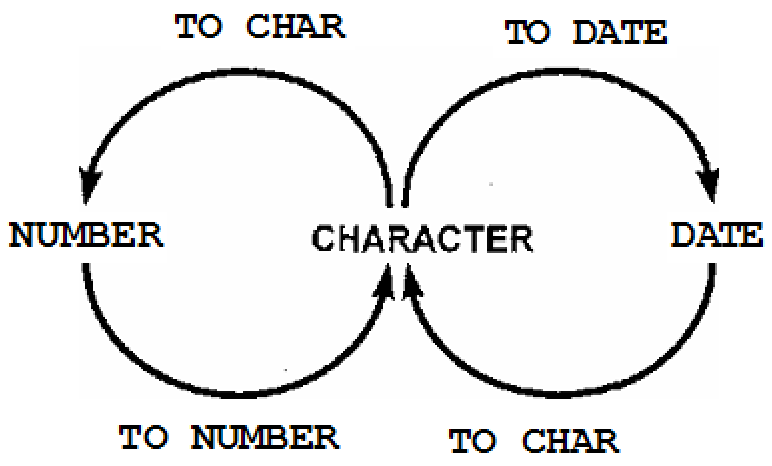
From	To
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

Implicit Datatype Conversion

In general, the Oracle Server uses the rule for expression when a datatype conversion is needed in places not covered by a rule for assignment conversions.

Note: CHAR to NUMBER conversions succeed only if the character string represents a valid number. CHAR to DATE conversions succeed only if the character string has the default format DD-MON-YY.

Explicit Datatype Conversion



Three Main Functions

SQL provides three functions to convert a value from one datatype to another:

TO_CHAR (*number* | *date* *[, 'fmt']*)

Converts a number or a date value to a VARCHAR2 character string with format model *fmt.*

TO_NUMBER (*char* *[, 'fmt']*)

Converts a character string containing digits to a number with the optional format model *fmt.*

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

Converts a character string representing a date to a date value according to the *fmt* specified (If *fmt* is omitted, format is DD-MON-YY.)

TO_CHAR (*number [,'fmt'] *)

TO_CHAR (number) converts *n* of NUMBER datatype to a value of VARCHAR2 datatype, using the optional number format *fmt*. If you omit *fmt*, then *n* is converted to a VARCHAR2 value exactly long enough to hold its significant digits.

The *'nlsparam'* specifies these characters that are returned by number format elements:

- Decimal character
- Group separator
- Local currency symbol
- International currency symbol

This argument can have this form:

```
'NLS_NUMERIC_CHARACTERS = ''dg''
   NLS_CURRENCY = ''text''
   NLS_ISO_CURRENCY = territory '
```

The characters ***d*** and ***g*** represent the decimal character and group separator, respectively. They must be different single-byte characters. Note that within the quoted string, you must use two single quotation marks around the parameter values. Ten characters are available for the currency symbol.

Examples

The following statement uses implicit conversion to interpret a string and a number into a number:

```
SELECT TO_CHAR('01110' + 1)
FROM dual;
```

TO_CHAR ('011')
1111

In the next example, the output is blank padded to the left of the currency symbol.

```
SELECT TO_CHAR(- 10000,'L99G999D99MI') "Amount"
     FROM DUAL;
```

Amount
TL10.000,00-

```
SELECT TO_CHAR(-10000,'L99,999.99MI') "Amount"
     FROM DUAL;
```

Amount
TL10,000.00-

```
SELECT TO_CHAR(-10000,'$99,999.99') "Amount"
     FROM DUAL;
```

Amount
-$10,000.00

```
SELECT TO_CHAR(-10000,'L99G999D99MI',
   'NLS_NUMERIC_CHARACTERS = '',.''
   NLS_CURRENCY = ''AusDollars'' ') "Amount"
     FROM DUAL;
```

Amount
AusDollars10.000,00-

```
SELECT TO_CHAR(-10000,'L99G999D99MI',
   'NLS_NUMERIC_CHARACTERS = '',.''
   NLS_CURRENCY = ''YTL'' ') "Miktar"
  FROM DUAL;
```

Miktar
YTL10.000,00-

```
SELECT TO_CHAR(-10000,'L99G999D99MI',
   'NLS_NUMERIC_CHARACTERS = '',.''
   NLS_CURRENCY = ''Yeni Türk Lirası'' ') "Miktar"
  FROM DUAL;
```

Miktar
Yeni Türk 10.000,00-

TO_CHAR Function with Dates

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

The format model:

·Must be enclosed in single quotation marks and is case sensitive

·Can include any valid date format element

·Has an *fm* element to remove padded blanks or suppress leading zeros

·Is separated from the date value by a comma

Displaying a Date **in a Specific** Format

- *Date Conversion Functions* are treated in a seperate chapter (see, *Date Conversion Functions* .)

TO_CHAR Function with Numbers

TO_CHAR (number, *' fmt')*

Use these formats with the TO_CHAR function to display a number.

9	Represents a number
0	Forces a zero to be displayed
$	Places a floating dollar sign
L	Uses the floating local currency symbol
.	Prints a decimal point
,	Prints a thousand indicator

TO_CHAR Function with Numbers

When working with number values such as character strings you should convert those numbers to the character datatype using the TO_CHAR function, which translates a value of NUMBER datatype to VARCHAR2 datatype. This Techniquc is especially useful *with* concatenation.

Using TO_CHAR Function with Numbers

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

SALARY
$3,000

Guidelines

The Oracle Server displays a string of pound signs (#) in place of a whole number whose digits exceed the number of digits provided in the format model.

The Oracle Server rounds the stored decimal value to the number of decimal spaces provided in the format model.

TO_NUMBER and TO_DATE Functions

Convert a character string to a number format using the TO_NUMBER function

TO_NUMBER(*char[,   'fmt']*)

- **Convert a character string to a date format using the TO_DATE function**

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

TO_NUMBER and TO_DATE Functions

You may want to convert a character string to either a number or a date. To accomplish this task, you use the TO_NUMBER or TO_DATE functions. The format model you choose will be based on the previously demonstrated format elements.

Example

Display the names and hiredates of all the employees who joined on February 22. 1981.

```
SELECT ename, hiredate
FROM   emp
WHERE  hiredate = TO_DATE ( 'Şubat 22, 1981', 'Month dd,  YYYY');
```

ENAME	HIREDATE
WARD	22/02/1981

NVL Function

Converts null to an actual value

- **Datatypes that can be used are date, character, and number,**
- **Datatypes must match**

-NVL(comm,0)

-NVL (hiredate, ‘01-JAN-97’)

-NVL (job , ‘No Job Yet’)

The NVL Function

To convert a null value to an actual value, use the NVL function.

Syntax

```
NVL ( expr-1 , expr-2)
```

expr-1	*is* the source value or expression that may contain null
expr-2	is the target value for converting null

You can use the NVL function to convert any datatype, but the return value is always the same as the datatype *of expr-1.*

NVL Conversions for Various Datatypes

Datatype	Conversion Example
NUMBER	NVL (number-column , 9)
DATE	NVL (date-column , ’01-JAN-95’)
CHAR or VARCHAR2	NVL (character-column , ‘Unavailable’)

If the NVL Function is Not Used

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

ENAME	SAL	COMM	(SAL*12)+COMM
SMITH	800		
ALLEN	1600	300	19500
WARD	1250	500	15500
JONES	2975		
MARTIN	1250	1400	16400
BLAKE	2850		
CLARK	2450		
SCOTT	3000		
KING	5000		
TURNER	1500	0	18000
ADAMS	1100		
JAMES	950		
FORD	3000		
MILLER	1300		

14 rows selected.

Using the NVL Function

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

ENAME	SAL	COMM	(SAL*12)+NVL(COMM,0)
SMITH	800		9600
ALLEN	1600	300	19500
WARD	1250	500	15500
JONES	2975		35700
MARTIN	1250	1400	16400
BLAKE	2850		34200
CLARK	2450		29400
SCOTT	3000		36000
KING	5000		60000
TURNER	1500	0	18000
ADAMS	1100		13200
JAMES	950		11400
FORD	3000		36000
MILLER	1300		15600

14 rows selected.

DECODE Function

Facilitates conditional inquiries by doing the work of a

CASE

or

IF-THEN-ELSE

statement

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

The DECODE Function

The DECODE function decodes an expression in a way similar to the **IF-THEN-ELSE** logic used in various languages. The DECODE function decodes *expression* after comparing it to each *search* value. If the expression is the same as ***search, result*** is returned.

If the default value is omitted. a null value is retuned where a search value does not match any of the result values.

Using the DECODE Function

```
SELECT   job, sal,
    DECODE(job,
                'ANALYST' , sal*1.1,
                'CLERK'   , sal*1.15,
                'MANAGER' , sal*1.20,
            sal)  ZAMLI_ÜCRETLER
FROM emp ;
```

JOB	SAL	ZAMLI_ÜCRETLE
CLERK	800	920
SALESMAN	1600	1600
SALESMAN	1250	1250
MANAGER	2975	3570
SALESMAN	1250	1250
MANAGER	2850	3420
MANAGER	2450	2940
ANALYST	3000	3300
PRESIDENT	5000	5000
SALESMAN	1500	1500
CLERK	1100	1265
CLERK	950	1092,5
ANALYST	3000	3300
CLERK	1300	1495

14 rows selected.

Using the DECODE Function

In the SQL statement above, the value of JOB is decoded. If JOB is ANALYST, the ssalary increase is 10% ; if JOB is CLERK, the salary increase is 15% , if JOB is MANAGER, the salary increase is 20%. For all other job roles, there is no increase in salary.

The same statement can be written as an IF-THEN-ELSE statement.

Using the DECODE Function

Display the applicable tax rate for each employee in department 30.

```
SELECT  ename, sal,
        DECODE (TRUNC (sal/1000, 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  emp ;
```

ENAME	SAL	TAX_RATE
ALLEN	1600	,09
WARD	1250	,09
MARTIN	1250	,09
BLAKE	2850	,2
TURNER	1500	,09
JAMES	950	0

6 rows selected.

Example

The slide shows another example using the DECODE function. In this example, we determine the tax rate for each employee in department 30 based upon the monthly salary. The tax rate is as follows:

Monthly Salary Range	*Rate*
`$ 0.00 - 999.99`	`0%`
`$1,000.00 - 1.999.99`	`9%`
`$2,000.00 - 2.999.99`	`20%`
`$3,000.00 - 3.999.99`	`30%`
`$4,000.00 - 4.999.99`	`40%`
`$5,000.00 - 5.999.99`	`42%`
`$6,000.00 - 6.999.99`	`44%`
`$7,000.00 or greater`	`45%`

Nesting Functions

Single-row functions can be nested to any level.

Nested functions are evaluated from deepest level to the least-deep level.

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

F1	: Step 1 = Result-1
F2	: Step 2= Result-2
F3	: Step 3 = Result-3

Nesting Functions

Single-row functions can be nested to any depth. Nested functions are evaluated from the intermost leve to the outermost level. Some examples follow to show you the flexibility of these functions.

Nesting Functions

```
SELECT    ename,
          NVL (TO_CHAR ( mgr )  , 'No Manager' )
FROM  emp
WHERE  mgr   IS   NULL;
```

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

Nesting Functions (continued)

The slide example displays the head of the companv, who has no manager. The evaluation of the SQL stalement involves two steps:

l. Evaluate the inner funclion to convert a number value to a character string

- Resultl =TO_CHAR (mgr)

2. Evaluate the outer function to replace the null value with a text string.

- NVL (Result1 , ‘No Manager')

The entire expression becomes the column heading because no column alias was given.

Example

Display the date of the next Friday that is six months from the hiredate. The resultant date should appearas Friday, March 12th 1982. Order the results by hiredate.

```
SELECT
TO_CHAR (NEXT_DAY (ADD_MONTHS (hiredate, 6),
'CUMA') , 'fmDay, Month ddth, YYYY' )
"Next 6 Months Review"
FROM emp
ORDER BY hiredate ;
```

Next 6 Months Review
Cuma, Haziran 19th, 1981
Cuma, Ağustos 21st, 1981
Cuma, Ağustos 28th, 1981
Cuma, Ekim 9th, 1981
Cuma, Kasım 6th, 1981
Cuma, Aralık 11th, 1981
Cuma, Mart 12th, 1982
Cuma, Nisan 2nd, 1982
Cuma, Mayıs 21st, 1982
Cuma, Haziran 4th, 1982
Cuma, Haziran 4th, 1982
Cuma, Temmuz 30th, 1982
Cuma, Haziran 10th, 1983
Cuma, Temmuz 15th, 1983

14 rows selected.

Summary

Use functions to do the following:

·Perform calculations on data

•Modify individual data items

•Manipulate output for groups of rows

•Alter date formats for display

·Convert column datatypes

Single- Row Functions

Single-row functions can be nested to any level. Single-row functions can manipulate the following

- Character data:

LOWER, UPPER, INITCAP, CONCAT, SUBSTR, INSTR, LENGTH

- Number data:

ROUND, TRUNC, MOD

- Date data:

MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND, TRUNC

- Date values can also use arithmetic operators.
- Conversion functions can convert character, date, and numeric values TO_CHAR, TO_DATE, TO_NUMBER

SYSDATE and DUAL

SYSDATE is a date function that returns the current date and time. It is customary to select SYSDATE from a dummy table called DUAL.