Datatype Comparison Rules

This section describes how Oracle Database compares values of each datatype.

Numeric Values

A larger value is considered greater than a smaller one. All negative numbers are less than zero and all positive numbers. Thus, -1 is less than 100; -100 is less than -1.

The floating-point value `NaN` (not a number) is greater than any other numeric value and is equal to itself.

See Also:

"Numeric Precedence " and "Floating-Point Numbers " for more information on comparison semantics

---

Date Values

A later date is considered greater than an earlier one. For example, the date equivalent of '29-MAR-1997' is less than that of '05-JAN-1998' and '05-JAN-1998 1:35pm' is greater than '05-JAN-1998 10:09am'.

Character String Values

Character values are compared using one of these comparison rules:

- Blank-padded comparison semantics
- Nonpadded comparison semantics

The following sections explain these comparison semantics.

Blank-Padded Comparison Semantics

If the two values have different lengths, then Oracle first adds blanks to the end of the shorter one so their lengths are equal. Oracle then compares the values character by character up to the first character that differs. The value with the greater character in the first differing position is considered greater. If two values have no differing characters, then they are considered equal. This rule means that two values are equal if they differ only in the number of trailing blanks. Oracle uses blank-padded comparison semantics only when both values in the comparison are either expressions of datatype `CHAR`, `NCHAR`, text literals, or values returned by the `USER` function.

Nonpadded Comparison Semantics

Oracle compares two values character by character up to the first character that differs. The value with the greater character in that position is considered greater. If two values of different length are identical up to the end of the shorter one, then the longer value is considered greater. If two values of equal length have no differing characters, then the values are considered equal. Oracle uses nonpadded comparison semantics whenever one or both values in the comparison have the datatype `VARCHAR2` or `NVARCHAR2`.

The results of comparing two character values using different comparison semantics may vary. The table that follows shows the results of comparing five pairs of character values using each comparison semantic. Usually, the results of blank-padded and nonpadded comparisons are the same. The last comparison in the table illustrates the differences between the blank-padded and nonpadded comparison semantics.

Blank-Padded	Nonpadded
`'ac' > 'ab'`	`'ac' > 'ab'`
`'ab' > 'a '`	`'ab' > 'a   '`
`'ab' > 'a'`	`'ab' > 'a'`
`'ab' = 'ab'`	`'ab' = 'ab'`
`'a ' = 'a'`	`'a ' > 'a'`

Single Characters

Oracle compares single characters according to their numeric values in the database character set. One character is greater than another if it has a greater numeric value than the other in the character set. Oracle considers blanks to be less than any character, which is true in most character sets.

These are some common character sets:

- 7-bit ASCII (American Standard Code for Information Interchange)
- EBCDIC Code (Extended Binary Coded Decimal Interchange Code)
- ISO 8859/1 (International Standards Organization)

- JEUC Japan Extended UNIX

Portions of the ASCII and EBCDIC character sets appear in Table 2-9 and Table 2-10. Uppercase and lowercase letters are not equivalent. The numeric values for the characters of a character set may not match the linguistic sequence for a particular language.

Table 2-9 ASCII Character Set

Symbol	Decimal value	Symbol	Decimal value
`blank`	`32`	`;`	`59`
`!`	`33`	`<`	`60`
`"`	`34`	`=`	`61`
`#`	`35`	`>`	`62`
`$`	`36`	`?`	`63`
`%`	`37`	`@`	`64`
`&`	`38`	`A-Z`	`65-90`
`'`	`39`	`[`	`91`
`(`	`40`	`\`	`92`
`)`	`41`	`]`	`93`
`*`	`42`	`^`	`94`
`+`	`43`	`_`	`95`
`,`	`44`	`‘`	`96`
`-`	`45`	`a-z`	`97-122`
`.`	`46`	`{`	`123`
`/`	`47`	`|`	`124`
`0-9`	`48-57`	`}`	`125`
`:`	`58`	`~`	`126`

Table 2-10 EBCDIC Character Set

Symbol	Decimal value	Symbol	Decimal value
`blank`	`64`	`%`	`108`
`¢`	`74`	`_`	`109`
`.`	`75`	`>`	`110`
`<`	`76`	`?`	`111`
`(`	`77`	`:`	`122`
`+`	`78`	`#`	`123`
`|`	`79`	`@`	`124`
`&`	`80`	`'`	`125`
`!`	`90`	`=`	`126`
`$`	`91`	`"`	`127`
`*`	`92`	`a-i`	`129-137`
`)`	`93`	`j-r`	`145-153`
`;`	`94`	`s-z`	`162-169`
`¬`	`95`	`A-I`	`193-201`
`-`	`96`	`J-R`	`209-217`
`/`	`97`	`S-Z`	`226-233`

Object Values

Object values are compared using one of two comparison functions: `MAP` and `ORDER`. Both functions compare object type instances, but they are quite different from one another. These functions must be specified as part of any object type that will be compared with other object types.

See Also:

CREATE TYPE for a description of `MAP` and `ORDER` methods and the values they return

Varrays and Nested Tables

You cannot compare varrays. Comparison of nested tables is described in "Comparison Conditions ".

Datatype Precedence

Oracle uses datatype precedence to determine implicit datatype conversion, which is discussed in the section that follows. Oracle datatypes take the following precedence:

- Datetime and interval datatypes
- `BINARY_DOUBLE`
- `BINARY_FLOAT`
- `NUMBER`

- Character datatypes
- All other built-in datatypes

Data Conversion

Generally an expression cannot contain values of different datatypes. For example, an expression cannot multiply 5 by 10 and then add 'JAMES'. However, Oracle supports both implicit and explicit conversion of values from one datatype to another.

Implicit and Explicit Data Conversion

Oracle recommends that you specify explicit conversions, rather than rely on implicit or automatic conversions, for these reasons:

- SQL statements are easier to understand when you use explicit datatype conversion functions.
- Implicit datatype conversion can have a negative impact on performance, especially if the datatype of a column value is converted to that of a constant rather than the other way around.
- Implicit conversion depends on the context in which it occurs and may not work the same way in every case. For example, implicit conversion from a datetime value to a `VARCHAR2` value may return an unexpected year depending on the value of the `NLS_DATE_FORMAT` parameter.
- Algorithms for implicit conversion are subject to change across software releases and among Oracle products. Behavior of explicit conversions is more predictable.

Implicit Data Conversion

Oracle Database automatically converts a value from one datatype to another when such a conversion makes sense. Table 2-11 is a matrix of Oracle implicit conversions. The table shows all possible conversions, without regard to the direction of the conversion or the context in which it is made. The rules governing these details follow the table.

Table 2-11 Implicit Type Conversion Matrix

	CHAR	VARCHAR2	NCHAR	NVARCHAR	DATE	DATETIME/INTERVAL	NUMBER	BINARY_FLOAT	BINARY_DOUBLE	LONG
CHAR	—	X	X	X	X	X	X	X	X	X
VARCHAR2	X	—	X	X	X	X	X	X	X	X
DATE	X	X	X	X	—	—	—	—	—	—
DATETIME/ INTERVAL	X	X	X	X	—	—	—	—	—	X
LONG	X	X	X	X	—	X	—	—	—	—
NUMBER	X	X	X	X	—	—	—	X	X	—
RAW	X	X	X	X	—	—	—	—	—	X
ROWID	X	X	X	X	—	—	—	—	—	—
CLOB	X	X	X	X	—	—	—	—	—	X
BLOB	—	—	—	—	—	—	—	—	—	—
NCHAR	X	X	—	X	X	X	X	X	X	X
NVARCHAR2	X	X	X	—	X	X	X	X	X	X
NCLOB	X	X	X	X	—	—	—	—	—	X
BINARY_FLOAT	X	X	X	X	—	—	X	—	X	—
BINARY_DOUBLE	X	X	X	X	—	—	X	X	—	—

The following rules govern the direction in which Oracle Database makes implicit datatype conversions:

- During `INSERT` and `UPDATE` operations, Oracle converts the value to the datatype of the affected column.
- During `SELECT FROM` operations, Oracle converts the data from the column to the type of the target variable.
- When comparing a character value with a numeric value, Oracle converts the character data to a numeric value.
- Conversions between character values or `NUMBER` values and floating-point number values can be inexact, because the character types and `NUMBER` use decimal precision to represent the numeric value, and the floating-point numbers use binary precision.
- Conversions from `BINARY_FLOAT` to `BINARY_DOUBLE` are exact.
- Conversions from `BINARY_DOUBLE` to `BINARY_FLOAT` are inexact if the `BINARY_DOUBLE` value uses more bits of precision that supported by the `BINARY_FLOAT`.
- When comparing a character value with a `DATE` value, Oracle converts the character data to `DATE`.
- When you use a SQL function or operator with an argument of a datatype other than the one it accepts, Oracle converts the argument to the accepted datatype.
- When making assignments, Oracle converts the value on the right side of the equal sign (=) to the datatype of the target of the assignment on the left side.
- During concatenation operations, Oracle converts from noncharacter datatypes to `CHAR` or `NCHAR`.
- During arithmetic operations on and comparisons between character and noncharacter datatypes, Oracle converts from any character datatype to a numeric, date, or rowid, as appropriate. In arithmetic operations between `CHAR`/`VARCHAR2` and `NCHAR`/`NVARCHAR2`, Oracle converts to a `NUMBER`.
- Comparisons between `CHAR` and `VARCHAR2` and between `NCHAR` and `NVARCHAR2` types may entail different character sets. The default direction of conversion in such cases is from the database character set to the national character set. Table 2-12 shows the direction of implicit conversions between different character types.

- Most SQL character functions are enabled to accept CLOBs as parameters, and Oracle performs implicit conversions between CLOB and character types. Therefore, functions that are not yet enabled for CLOBs can accept CLOBs through implicit conversion. In such cases, Oracle converts the CLOBs to CHAR or VARCHAR2 before the function is invoked. If the CLOB is larger than 4000 bytes, then Oracle converts only the first 4000 bytes to CHAR.

Table 2-12 Conversion Direction of Different Character Types

	to CHAR	to VARCHAR2	to NCHAR	to NVARCHAR2
from CHAR	—	VARCHAR2	NCHAR	NVARCHAR2
from VARCHAR2	VARCHAR2	—	NVARCHAR2	NVARCHAR2
from NCHAR	NCHAR	NCHAR	—	NVARCHAR2
from NVARCHAR2	NVARCHAR2	NVARCHAR2	NVARCHAR2	—

Implicit Data Conversion Examples

Text Literal Example

The text literal '10' has datatype CHAR. Oracle implicitly converts it to the NUMBER datatype if it appears in a numeric expression as in the following statement:

```
SELECT salary + '10'
   FROM employees;
```

Character and Number Values Example

When a condition compares a character value and a NUMBER value, Oracle implicitly converts the character value to a NUMBER value, rather than converting the NUMBER value to a character value. In the following statement, Oracle implicitly converts '200' to 200:

```
SELECT last_name
    FROM employees
    WHERE employee_id = '200';
```

Date Example

In the following statement, Oracle implicitly converts '03-MAR-97' to a DATE value using the default date format 'DD-MON-YY':

```
SELECT last_name
    FROM employees
    WHERE hire_date = '03-MAR-97';
```

Rowid Example

In the following statement, Oracle implicitly converts the text literal 'AAAFYmAAFAAAAFGAAH' to a rowid value. (Rowids are unique within a database, so to use this example you must know an actual rowid in your database.)

```
SELECT last_name
    FROM employees
    WHERE ROWID = 'AAAFd1AAFAAAABSAAH';
```

Explicit Data Conversion

You can explicitly specify datatype conversions using SQL conversion functions. Table 2-13 shows SQL functions that explicitly convert a value from one datatype to another.

You cannot specify LONG and LONG RAW values in cases in which Oracle can perform implicit datatype conversion. For example, LONG and LONG RAW values cannot appear in expressions with functions or operators. Please refer to "LONG Datatype " for information on the limitations on LONG and LONG RAW datatypes.

Table 2-13 Explicit Type Conversions

	to CHAR,VARCHAR2,NCHAR,NVARCHAR2	to NUMBER	to Datetime/Interval	to RAW	to ROWID	to LONG,LONG RAW	to CLOB, NCLOB,BLOB	to BINARY
from CHAR, VARCHAR2, NCHAR, NVARCHAR2	TO_CHAR (char.) TO_NCHAR (char.)	TO_NUMBER	TO_DATE TO_TIMESTAMP TO_TIMESTAMP_TZ TO_YMINTERVAL TO_DSINTERVAL	HEXTORAW	CHARTO-=ROWID	–	TO_CLOB TO_NCLOB	TO_BINAR
from NUMBER	TO_CHAR (number) TO_NCHAR (number)	–	TO_DATE NUMTOYM- INTERVAL NUMTODS- INTERVAL	–	–	–	–	TO_BINAR
from Datetime/ Interval	TO_CHAR (date) TO_NCHAR (datetime)	–	–	–	–	–	–	–
from RAW	RAWTOHEX RAWTONHEX	–	–	–	–	–	TO_BLOB	–

	to CHAR,VARCHAR2,NCHAR,NVARCHAR2	to NUMBER	to Datetime/Interval	to RAW	to ROWID	to LONG,LONG RAW	to CLOB, NCLOB,BLOB	to BINARY
from ROWID	`ROWIDTOCHAR`	–	–	–	–	–	–	–
from LONG / LONG RAW	–	–	–	–	–	–	`TO_LOB`	–
from CLOB, NCLOB, BLOB	`TO_CHAR` `TO_NCHAR`	–	–	–	–	–	`TO_CLOB` `TO_NCLOB`	–
from CLOB, NCLOB, BLOB	`TO_CHAR` `TO_NCHAR`	–	–	–	–	–	`TO_CLOB` `TO_NCLOB`	–
from BINARY_FLOAT	`TO_CHAR (char.)` `TO_NCHAR (char.)`	`TO_NUMBER`	–	–	–	–	–	`TO_BINAR`
from BINARY_DOUBLE	`TO_CHAR (char.)` `TO_NCHAR (char.)`	`TO_NUMBER`	–	–	–	–	–	`TO_BINAR`

See Also:

"Conversion Functions " for details on all of the explicit conversion functions