SQL> select \* from voucher\_series;

no rows selected

SQL> insert into voucher\_series(valid\_from,valid\_to)

2 values( to\_timestamp\_tz(' 18 Feb 2010 08:37:00 EST',' DD Mon Y

SS TZD') ,

3 to\_timestamp\_tz(' 2012-10-01 00:00:00+02')); // NO FORMATE NEED TO SPECIFY WHILE INSERTION.

1 row created.

'2020-11-30 00:00:00.561+01',' YYYY-MM-DD HH24:MI:SS.**FF** TZH' FF for precision after sec

SQL> select \* from voucher\_series;

ID SERIES VALID\_FROM VALID\_TO PARTNER UNITS

2010-02-18 08:37:00+05 2012-10-01 00:00:00+02

DISPLAYING format is *just a matter of presentation* . Run below command before “select \*”

alter session set NLS\_TIMESTAMP\_TZ\_FORMAT =

'YYYY-MM-DD HH24:MI:SSTZH';

More format options <http://docs.oracle.com/cd/E11882_01/server.112/e10729/ch3globenv.htm#g1012703>

<http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1029.htm#BABEFHFG>

-----------------------------------------------------------------------------------------------

|  |  |
| --- | --- |
|  | Date columns don't have timezone as an option. You'd have to create the column as data type TIMESTAMP WITH TIME ZONEorTIMESTAMP WITH LOCAL TIME ZONE, and besides, the TO\_DATE function doesn't understand the TIME ZONE format mask you're applying. |

|  |  |
| --- | --- |
| 4 down vote accepted | Changing the date format is *just a matter of presentation*. When selecting from you database, apply a format:  select to\_char(sysdate, 'YYYY-MM-DD') from dual;  To get this format by default, set it in your session’s NLS\_DATE\_FORMAT parameter:  alter session set NLS\_DATE\_FORMAT = 'YYYY-MM-DD';  You can also set the NLS\_TIMESTAMP\_FORMAT and NLS\_TIMESTAMP\_TZ\_FORMAT. See [Setting Up a Globalization Support Environment](http://docs.oracle.com/cd/E11882_01/server.112/e10729/ch3globenv.htm#g1012703) in Oracle’s documentation.  Regarding time zones, normally it is the application that inserts the timestamp with time zone in the database that chooses what time zone it puts. But the application that queries these timestamps can request to get them back in another time zone. Compare the following queries:  select systimestamp from dual; -- This gives you the timestamp in the server configured time zone  select systimestamp at time zone 'EST' from dual; -- This is in your EST time zone  select systimestamp at time zone '+00:00' from dual; -- Finally the same timestamp in UTC.  To force a certain time zone you can force it with an after insert or update trigger. Something like:  create or replace trigger my\_table\_force\_utc\_timestamps  before insert or update on my\_table  for each row  begin  :new.my\_timestamp\_with\_tz := :new.my\_timestamp\_with\_tz at time zone '+00:00';  end;  /  If you *really* want to convert your timestamps to UTC you can try:  update my\_table set my\_timestamp\_with\_tz = my\_timestamp\_with\_tz at time zone '+00:00';  *Edit*: to change sysdate time zone before inserting into tables you must in fact use systimestamp at time zone … that will be cast to a date value:  insert into my\_table (date\_column) values (systimestamp at time zone '+00:00');  You can read more about time zones in Oracle’s documentation [Datetime and Time Zone Parameters and Environment Variables](http://docs.oracle.com/cd/E11882_01/server.112/e10729/ch4datetime.htm#i1006532). With the example table\_dt this would give: |

===============================

TO\_TIMESTAMP\_TZ

<http://www.java2s.com/Book/Oracle/Timestamp_Functions/TO_TIMESTAMP_TZ.htm>

Difference beween Timestamp , timestamp\_tz, TIMESTAMP\_LTZ, Date tepy descriptions

<http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#i1020862>

**OLAP DML Data Types**

In the OLAP DML, as in other languages, a data type is a collection of values and the definition of one or more operations on those values.

The Oracle OLAP DML supports the data types outlined in [Table 2-1, "Summary of OLAP DML Data Types"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJADHGII).

Table 2-1 Summary of OLAP DML Data Types

| **Data Type** | **Abbreviation** | **Description** |
| --- | --- | --- |
| BOOLEAN | BOOL | Represents the logical TRUE and FALSE values. |
| DATE | None | Does not correspond to the SQL data type of the same name; but, instead, is an older data type that is unique to the OLAP DML.  Day, month, and year data (but not hour and minute data) between January 1, 1000 A.D. and December 31, 9999 A.D. |
| DATETIME | None | Corresponds to the SQL DATE data type.  Valid date range from January 1, 4712 BC to December 31, 9999 AD. The default format is determined explicitly by the NLS\_DATE\_FORMAT parameter or implicitly by the NLS\_TERRITORY parameter. The size is fixed at 7 bytes. This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It does not have fractional seconds or a time zone. |
| TIMESTAMP | None | Corresponds to the SQL TIMESTAMP data type.  Year, month, and day values of date, as well as hour, minute, and second values of time up to a precision of 9 places for the fractional part of the SECOND datetime field. The default format is determined explicitly by the NLS\_DATE\_FORMAT parameter or implicitly by the NLS\_TERRITORY parameter. The size is 11 bytes. This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, and SECOND. It contains fractional seconds but does not have a time zone. |
| TIMESTAMP\_TZ | None | Corresponds to the SQL TIMESTAMP WITH TIME ZONE data type.  All values of TIMESTAMP as well as time zone displacement value, with a precision of 9 places for the fractional part of the SECOND datetime field. The default format is determined explicitly by the NLS\_DATE\_FORMAT parameter or implicitly by the NLS\_TERRITORY parameter. The size is fixed at 13 bytes. This data type contains the datetime fields YEAR, MONTH, DAY, HOUR, MINUTE, SECOND, TIMEZONE\_HOUR, and TIMEZONE\_MINUTE. It has fractional seconds and an explicit time zone. |
| TIMESTAMP\_LTZ |  | Corresponds to the SQL TIMESTAMP WITH LOCAL TIME ZONE data type.  All values of TIMESTAMP\_TZ, with the following exceptions:   * Data is normalized to the Database time zone when it is stored in the Database. * When the data is retrieved, users see the data in the session time zone.   The default format is determined explicitly by the NLS\_DATE\_FORMAT parameter or implicitly by the NLS\_TERRITORY parameter. The size is 11 bytes. |
| DSINTERVAL | None | Corresponds to the SQL INTERVAL DAY TO SECOND data type.  Stores a period of time in days, hours, minutes, and seconds. |
| YMINTERVAL | None | Corresponds to the SQL INTERVAL YEAR TO MONTH data type.  Stores a period of time in years and months. |
| INTEGER | INT | A whole number in the range of (-2\*\*31) to (2\*\*31)-1. |
| SHORTINTEGER | SHORTINT | A whole number in the range of (-2\*\*15) to (2\*\*15)-1. |
| LONGINTEGER | LONGINT | A whole number in the range of (-2\*\*63) to (2\*\*63)-1. |
| DECIMAL | DEC | A decimal number with up to 15 significant digits in the range of -(10\*\*308) to +(10\*\*308). |
| SHORTDECIMAL | SHORT | A decimal number with up to 7 significant digits in the range of -(10\*\*38) to +(10\*\*38). |
| NUMBER [(p,[s])] | None | A decimal number with up to 38 significant digits in the range of -(10\*\*125) to +(10\*\*125). |
| TEXT | None | Up to 4000 bytes for each line in the Database character set. This data type is equivalent to the CHAR and VARCHAR2 data types in the Database. (Note that when defining a variable of this data type you specify the RANSPACE64 keyword in the DEFINE VARIABLE statement to increase the maximum number of characters for the values of the variable from nearly 2\*\*32 to nearly 2\*\*64.) |
| NTEXT | None | Up to 4000 bytes for each line in UTF-8 character encoding. This data type is equivalent to the NCHAR and NVARCHAR2 data types in the Database. (Note that when defining a variable of this data type you specify the RANSPACE64 keyword in the DEFINE VARIABLE statement to increase the maximum number of characters for the values of the variable from nearly 2\*\*32 to nearly 2\*\*64.) |
| ID | None | Up to 8 single-byte characters for each line in the database character set. (ID is valid only for values of simple dimensions, see [DEFINE DIMENSION (simple)](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1031.htm#i1010637).) |
| RAW (size) | None | Raw binary data of length size bytes. Maximum size is 2000 bytes. You must specify size for a RAW value. (Note that when defining a variable of this data type you specify the RANSPACE64 keyword in the DEFINE VARIABLE statement to increase the maximum number of characters for the values of the variable from nearly 2\*\*32 to nearly 2\*\*64.) |
| ROWID | None. | Base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn. |
| UROWID | None | Base 64 string representing the logical address of a row of an index-organized table. The optional size is the size of a column of type UROWID. The maximum size and default is 4000 bytes. |
| WORKSHEET |  | Specified for arguments and temporary variables in an OLAP DML program when you want to handle arguments without converting values to a specific data type. Use the WKSDATA function to retrieve the data type of an argument with a WORKSHEET data type. |

Categories of Data Types

Frequently, these data types are thought of as belonging to the following categories:

* [Numeric Data Types](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJACDECG) which are INTEGER, SHORTINTEGER, LONGINTEGER, DECIMAL, SHORTDECIMAL, and NUMBER
* [Text Data Types](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#i1012270)which are TEXT, NTEXT and ID.
* [Datetime and Interval Data Types](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#BABJGJIH) which includes the datetime data types of DATETIME, TIMESTAMP, TIMESTAMP\_TZ, and TIMESTAMP-LTZ and the interval data types of DSINTERVAL and YMINTERVAL.
* [Date-only Data Type](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJAIFGIC) which is the DATE data type that is unique to the OLAP DML.
* [Boolean Data Type](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJACGBGE) which is BOOLEAN.
* [Row Identifier Data Types](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#BABBDCFA) which are ROWID and UROWID.

Which OLAP DML Data Objects Can Have Which Data Type?

Different objects support the use of different data types for their values:

* For variables, all of the data types are supported.
* For dimensions and surrogates, the INTEGER, NUMBER, TEXT, ID (simple dimensions only), NTEXT, DATETIME, TIMESTAMP, TIMESTAMP\_TZ, TIMESTAMP-LTZ, DSINTERVAL, and YMINTERVAL data types are supported. Additionally, when you define a dimension of type DAY, WEEK, MONTH, QUARTER, or YEAR using a [DEFINE DIMENSION (DWMQY)](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1031.htm#CHDIDHBI) statement, the data type of the values of that dimension are DATE-only.

**Numeric Data Types**

The numeric data types described in [Table 2-2, "OLAP DML Numeric Data Types"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#g1022512) are supported.

Table 2-2 OLAP DML Numeric Data Types

| **Data Type** | **Data Value** |
| --- | --- |
| INTEGER | A whole number in the range of (-2\*\*31) to (2\*\*31)-1. |
| SHORTINTEGER | A whole number in the range of (-2\*\*15) to (2\*\*15)-1. |
| LONGINTEGER | A whole number in the range of (-2\*\*63) to (2\*\*63)-1. |
| DECIMAL | A decimal number with up to 15 significant digits in the range of -(10\*\*308) to +(10\*\*308). |
| SHORTDECIMAL | A decimal number with up to 7 significant digits in the range of -(10\*\*38) to +(10\*\*38). |
| NUMBER | A decimal number with up to 38 significant digits in the range of -(10\*\*125) to +(10\*\*125). |

For data entry, a value for any of these data types can begin with a plus (+) or minus (-) sign; it cannot contain commas. Note, however, that a comma is required before a negative number that follows another numeric expression, or the minus sign is interpreted as a subtraction operator. Additionally, a decimal value can contain a decimal point. For data display, thousands and decimal markers are controlled by the NLS\_NUMERIC\_CHARACTERS option.

**Using LONGINTEGER Values**

Most of the numerical data types return NA when a value is outside its range. However, the LONGINTEGER data type does not have overflow protection and will return an incorrect value when, for example, a calculation produces a number that exceeds its range. Use the NUMBER data type instead of LONGINTEGER when this is likely to be a problem.

**Using NUMBER Values**

When you define a NUMBER variable, you can specify its precision (p) and scale (s) so that it is sufficiently, but not unnecessarily, large. Precision is the number of significant digits. Scale can be positive or negative. Positive scale identifies the number of digits to the right of the decimal point; negative scale identifies the number of digits to the left of the decimal point that can be rounded up or down.

The NUMBER data type is supported by Oracle Database standard libraries and operates the same way as it does in SQL. It is used for dimensions and surrogates when a text or INTEGER data type is not appropriate. It is typically assigned to variables that are not used for calculations (like forecasts and aggregations), and it is used for variables that must match the rounding behavior of the Database or require a high degree of precision. When deciding whether to assign the NUMBER data type to a variable, keep the following facts in mind in order to maximize performance:

* Analytic workspace calculations on NUMBER variables is slower than other numerical data types because NUMBER values are calculated in software (for accuracy) rather than in hardware (for speed).
* When data is fetched from an analytic workspace to a relational column that has the NUMBER data type, performance is best when the data already has the NUMBER data type in the analytic workspace because a conversion step is not required.

**Text Data Types**

The text data types described in [Table 2-3, "OLAP DML Text Data Types"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#g1022561) are supported by Oracle OLAP.

Table 2-3 OLAP DML Text Data Types

| **Data Type** | **Data Value** |
| --- | --- |
| TEXT | Up to 4000 bytes for each line in the database character set. This data type is equivalent to the CHAR and VARCHAR2 data types in the Database. |
| NTEXT | Up to 4000 bytes for each line in UTF-8 character encoding. This data type is equivalent to the NCHAR and NVARCHAR2 data types in the Database. |
| ID | Up to 8 single-byte characters for each line in the database character set. (ID is valid only for values of simple dimensions, see [DEFINE DIMENSION (simple)](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1031.htm#i1010637).) |

**Text Literals**

Enclose text literals in single quotes. Oracle OLAP recognizes unquoted alpha-numeric values as object names and double quotes as the beginning of a comment.

You can embed quoted strings within a quoted string, which is necessary when you want to specify the base dimension value of a composite or conjoint dimension or when a value includes an apostrophe. Since a single quotation mark is used in Oracle OLAP to indicate a text string, it is considered a special character when used within such a string. Consequently, to specify the literal value of a single quotation mark within a text string, precede the quotation mark with a backslash.

For example, suppose you want to find out if New York and Apple Sauce are a valid combination of base dimension values in the markprod conjoint dimension. The following statement produces the answer YES or NO.

SHOW ISVALUE(markprod, '<\'New York\' \'Apple Sauce\'>')

When embedded quoted strings have a further level of embedding, you must use backslashes before each special character, such as the apostrophe and the backslash that must precede it in "Joe's Deli," as shown in the following statement.

SHOW ISVALUE(markprod, '<\'Joe\\\'s Deli\' \'Apple Sauce\'>')

**Escape Sequences**

[Table 2-4, "Recognized Escape Sequences"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm" \l "g1022641) shows escape sequences that are recognized by Oracle OLAP.

Table 2-4 Recognized Escape Sequences

| **Sequence** | **Meaning** |
| --- | --- |
| \b | Backspace |
| \f | Form feed |
| \n | Line feed |
| \r | Carriage return |
| \t | Horizontal tab |
| \" | Double quote |
| \' | Single quote |
| \\ | Backslash |
| \dnnn | Character with ASCII code nnn decimal, where \d indicates a decimal escape and nnn is the decimal value for the character |
| \xnn | Character with ASCII code nn hexadecimal, where \x indicates a hexadecimal escape and nn is the hexadecimal value for the character |
| \Unnnn | Character with Unicode nnnn, where \U indicates a Unicode escape and nnnn is a four-digit hexadecimal INTEGER that represents the Unicode codepoint with the value U+nnnn. The U must be a capital letter. |

**Date-only Data Type**

The Oracle OLAP DML DATE data type does not correspond to the SQL data type of the same name. It is, instead, is an older data type that is unique to the OLAP DML. The OLAP DML DATE data type is a valid data type for variables and for dimensions of type DAY, WEEK, MONTH, QUARTER, and YEAR as discussed in the [DEFINE DIMENSION (DWMQY)](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1031.htm#CHDIDHBI) command topic. It is used to store day, month, and year data (but not hour and minute data) between January 1, 1000 A.D. and December 31, 9999 A.D. Because the OLAP DML DATE data type does not include hour and minute data, it is often referred to as the DATE-only data type.

Tip:

The Oracle OLAP DML data type that corresponds to the SQL DATE data type is named DATETIME. See [DATETIME Data Type](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJAHJAJH) for more information.

See also:

["Date-only Data Type Options"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_options003.htm#CHDEHCCI).

**Date-only Input Values**

A valid input literal value of type DATE must conform to one of three styles: numeric, packed numeric, or month name. You can mix these styles throughout a session.

Tip:

To determine whether a text expression (such as an expression with a data type of TEXT or ID) represents a valid DATE-only value, use the [ISDATE](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_functions_1106.htm#i79562) program

Numeric style

Specify the day, month, and year as three INTEGER values with one or more separators between them, using these rules:

* The day and month components can have one digit or two digits.
* For any year, the year component can have four digits (for example, 1997). For years in the range 1950 to 2049, the year component can, alternatively, have two digits (50 represents 1950, and so on).
* To separate the components, you can use a space, dash (-), slash (/), colon (:), or comma (,).

Examples: '24/4/97' or '24-04-1997'

Packed numeric style

Specify the day, month, and year as three INTEGER values with no separators between them, using these rules:

* The day and month components must have two digits. When the day or month is less than 10, it must be preceded by a zero.
* For any year, the year component can have four digits (for example, 1997). For years in the range 1950 to 2049, the year component can, alternatively, have two digits (50 represents 1950, and so on).
* You cannot use any separators between the date components.

Examples: '240497' or '04241997'

Month name style

Specify the day and year as INTEGER values and the month as text, using these rules:

* The month component must match one of the names listed in the MONTHNAMES option. You can abbreviate the month name to one letter or more, when you supply enough letters to uniquely match the beginning of a name in MONTHNAMES. The case of the letters in the month component (uppercase or lowercase) does not need to match the case in MONTHNAMES.
* The day component can have one digit or two digits.
* For any year, the year component can have four digits (for example, 1997). For years in the range 1950 to 2049, the year component can, alternatively, have two digits (50 represents 1950, and so on).
* When the day and year components are adjacent, they must have at least one separator between them. As separators, you can use a space, dash (-), slash (/), colon (:), or comma (,). When you want, you can place one or more separators between the day and month or between the year and month.

Examples: '24APR97' or '24 ap 97' or 'April 24, 1997'

**Date-only Dimension Values**

The format of a DATE -only value of a dimension of type DAY, WEEK, MONTH, QUARTER, or YEAR is determined by the value name format (VNF) associated with the object. A VNF is a template that controls the input and display format for DATE -only values. The template can include format specifications for any of the components that identify a time period (day, month, calendar year, fiscal year, and period within a fiscal year). You associate a VNF with an object by adding a VNF statement to its definition. When you do not add a VNF to the definition of an object, the object uses the default VNF shown in [Table 2-5, "Default VNFs for DWMQY Dimensions"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#g1022167).

Table 2-5 Default VNFs for DWMQY Dimensions

| **Type of Dimension** | **Default VNF** | **Example** |
| --- | --- | --- |
| DAY | <DD><MTXT><YY> | 01JAN95 |
| WEEK | W<P>.<FF> | W1.95 |
| Multiple WEEK | <NAME><P>.<FF> | MYWEEK1.95 |
| MONTH | <MTXT><YY> | JAN95 |
| Multiple MONTH | <NAME><P>.<FF> | MYMONTH1.95 |
| QUARTER | Q<P>.<FF> | Q1.95 |
| YEAR | YR<YY> | YR95 |

DATE-only values have independent input and output formats. You can enter DATE-only values in one style and report them in a different style.

**DATE-only Variable Display Styles**

When you show a DATE-only variable value in output, the format depends on the DATEFORMAT option. The default format is a 2-digit day, a 3-letter month, and a 2-digit year; for example, 03MAR97. The text for the month names depends on the MONTHNAMES option. To change the order of the month, day, and year components, see the [DATEORDER](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_options015.htm#i79233) option.

**Datetime and Interval Data Types**

The OLAP DML has data types that correspond to SQL datetime and interval data types. As outlined in [Table 2-6, "OLAP DML Datetime and Interval Data Types and the Corresponding SQL Data Types"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#g1022681), the names of the data types are different in OLAP DML than they are in SQL.

Table 2-6 OLAP DML Datetime and Interval Data Types and the Corresponding SQL Data Types

| **OLAP DML Data Type** | **Corresponding SQL Data Type** |
| --- | --- |
| DATETIME | DATE |
| TIMESTAMP | TIMESTAMP |
| TIMESTAMP\_TZ | TIMESTAMP WITH TIMEZONE |
| TIMESTAMP\_LTZ | TIMESTAMP WITH LOCAL TIME ZONE |
| DSINTERVAL | INTERVAL DAY TO SECOND |
| YMINTERVAL | INTERVAL YEAR TO MONTH |

In the OLAP DML, the datetime data types are DATETIME, TIMESTAMP, TIMESTAMP\_TZ, and TIMESTAMP\_LTZ. The interval data types are YMINTERVAL and DSINTERVAL. Both datetimes and intervals are made up of fields as discussed in ["Datetime and Interval Fields"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJAFCGGF).

Note:

The Oracle OLAP DML has a date data type named DATE that does not correspond to the SQL data type of that name. (The OLAP DML DATE data type was implement before the SQL datetime and interval data types were implemented in the OLAP DML.) The OLAP DML DATE data type stores only date values (no time values) and is therefore sometimes referred to as the DATE-only data type.

**Datetime and Interval Fields**

Both datetimes and intervals are made up of fields. The values of these fields determine the value of the data type. [Table 2-7, "Datetime Fields and Values"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#BABFDAEI) lists the datetime fields and their possible values for datetimes and intervals.

Tip:

To avoid unexpected results in your operations on datetime data, you can verify the Database and session time zones using DBTIMEZONE and SESSIONTIMEZONE If the time zones have not been set manually, Oracle Database uses the operating system time zone by default. If the operating system time zone is not a valid Oracle time zone, then Oracle uses UTC as the default value.

Table 2-7 Datetime Fields and Values

| **Datetime Field** | **Valid Values for Datetime Data Types** | **Valid Values for Interval Data Types** |
| --- | --- | --- |
| YEAR | -4712 to 9999 (excluding year 0) | Any positive or negative integer |
| MONTH | 01 to 12 | 0 to 11 |
| DAY | 01 to 31 (limited by the values of MONTH and YEAR, according to the rules of the current NLS calendar parameter) | Any positive or negative integer |
| HOUR | 00 to 23 | 0 to 23 |
| MINUTE | 00 to 59 | 0 to 59 |
| SECOND | 00 to 59.9(n), where 9(n) is the precision of time fractional seconds. The 9(n) portion is not applicable for DATETIME. | 0 to 59.9(n), where 9(n) is the precision of interval fractional seconds |
| TIMEZONE\_HOUR | -12 to 14 (This range accommodates daylight saving time changes.) Not applicable for DATETIME or TIMESTAMP. | Not applicable |
| TIMEZONE\_MINUTE  (See note at end of table) | 00 to 59. Not applicable for DATETIME or TIMESTAMP. | Not applicable |
| TIMEZONE\_REGION | Query the TZNAME column of the V$TIMEZONE\_NAMES data dictionary view. Not applicable for DATETIME or TIMESTAMP. For a complete listing of all timezone regions, refer to Oracle Database Globalization Support Guide. | Not applicable |
| TIMEZONE\_ABBR | Query the TZABBREV column of the V$TIMEZONE\_NAMES data dictionary view. Not applicable for DATETIME or TIMESTAMP. | Not applicable |

Note: TIMEZONE\_HOUR and TIMEZONE\_MINUTE are specified together and interpreted as an entity in the format +|- hh:mm, with values ranging from -12:59 to +14:00.

See Also:

["Datetime and Interval Expressions"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression003.htm#BABFFDGC)

**Datetime Format Templates**

A datetime format template is a template that describes the format of datetime data stored in a character string. A format model does not change the internal representation of the value in the Database. When you convert a character string into a date, a format model determines how Oracle Database interprets the string. In OLAP DML statements, you can use a format model as an argument of the TO\_CHAR and TO\_DATE functions to specify:

* The format for Oracle to use to return a value from the Database
* The format for a value you have specified for Oracle to store in the Database

You can use datetime format templates in the following functions:

* In the TO\_\* datetime functions to translate a character value that is in a format other than the default format into a datetime value. (The TO\_\* datetime functions are TO\_CHAR, TO\_DATE, TO\_TIMESTAMP, TO\_TIMESTAMP\_TZ, TO\_YMINTERVAL, and TO\_DSINTERVAL.)
* In the TO\_CHAR function to translate a datetime value that is in a format other than the default format into a string (for example, to print the date from an application)

The default datetime formats are specified either explicitly with the initialization parameter NLS\_DATE\_FORMAT or implicitly with the initialization parameter NLS\_TERRITORY. You can change the default datetime formats for your session with the ALTER SESSION statement. You can override this default and specify a datetime format for use with a particular OLAP DML object by using the [DATE\_FORMAT](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1029.htm#CACIIBBF) command to add a datetime format to the definition of the object.

**String-to-Date Conversion Rules**

The following additional formatting rules apply when converting string values to datetime values (unless you have used the FX or FXFM modifiers in the format model to control exact format checking):

* You can omit punctuation included in the format string from the date string if all the digits of the numerical format elements, including leading zeros, are specified. In other words, specify 02 and not 2 for two-digit format elements such as MM, DD, and YY.
* You can omit time fields found at the end of a format string from the date string.
* If a match fails between a datetime format element and the corresponding characters in the date string, then Oracle attempts alternative format elements, as shown in [Table 2-8, "Oracle Format Matching"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#g195479).

Table 2-8 Oracle Format Matching

| **Original Format Element** | **Additional Format Elements to Try in Place of the Original** |
| --- | --- |
| 'MM' | 'MON' and 'MONTH' |
| 'MON | 'MONTH' |
| 'MONTH' | 'MON' |
| 'YY' | 'YYYY' |
| 'RR' | 'RRRR' |

**DATETIME Data Type**

The OLAP DML DATETIME data type corresponds to the SQL DATE data type. As such, the format and language of DATETIME values are controlled by the settings of the NLS\_DATE\_FORMAT and NLS\_DATE\_LANGUAGE options. The DATETIME data type is supported by Oracle Database standard libraries and operates the same way in the OLAP DML as it does the DATE data type in SQL.

Note:

The Oracle OLAP DML has a date data type named DATE that does not correspond to the SQL data type of that name. The OLAP DML DATE data type stores only date values (no time values) and is therefore sometimes referred to as the DATE-only data type. The DATEORDER, DATEFORMAT, and MONTHNAMES options, which control the formatting of DATE values, have no effect on DATETIME values. However, DATETIME and DATE values can be used interchangeably in most DML statements.

You can specify a DATETIME value as a string literal, or you can convert a character or numeric value to a date value with the TO\_DATE function.

To specify a DATETIME value as a literal, you must use the Gregorian calendar. You can specify an ANSI literal, as shown in this example:

DATETIME '1998-12-25'

The ANSI date literal contains no time portion, and must be specified in exactly this format ('YYYY-MM-DD').

Alternatively you can specify a DATETIME value us the TO\_DATE function and include, as in the following example:

TO\_DATE('98-DEC-25 17:30','YY-MON-DD HH24:MI')

The default date format template for an Oracle DATETIME value is specified by the initialization parameter NLS\_DATE\_FORMAT. This example date format includes a two-digit number for the day of the month, an abbreviation of the month name, the last two digits of the year, and a 24-hour time designation.

Oracle automatically converts character values that are in the default datetime format into datetime values when they are used in datetime expressions.

If you specify a datetime value without a time component, then the default time is midnight (00:00:00 or 12:00:00 for 24-hour and 12-hour clock time, respectively). If you specify a datetime value without a date, then the default date is the first day of the current month.

Values of DATETIME always contain both the date and time fields. Therefore, if you use DATETIME values in an expression, you must either specify the time field in your query or ensure that the time fields in the DATETIME values are set to midnight. Otherwise, Oracle may not return the results you expect. You can use the TRUNC (date) function to set the time field to midnight, or you can include a greater-than or less-than condition in the query instead of an equality or inequality condition. However, if the expression contains DATETIME values other than midnight, then you must filter out the time fields in the query to get the correct result.

The date function SYSDATE returns the current system date and time. The function CURRENT\_DATE returns the current session date. For information on SYSDATE, the TO\_\* datetime functions, and the default date format, see ["Datetime functions"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_functions_1003.htm#CJAEDBFF) and the [DATE\_FORMAT](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_1029.htm#CACIIBBF) command.

**TIMESTAMP Data Type**

The TIMESTAMP data type is an extension of the DATETIME data type. It stores the year, month, and day of the DATETIME data type, plus hour, minute, and second values. This data type is useful for storing precise time values.

SHOW SYSDATE

26-JUL-06

DEFINE mytimestamp VARIABLE TIMESTAMP

mytimestamp = SYSDATE

COLWIDTH = 30

REPORT mytimestamp

MYTIMESTAMP

------------------------------

26-JUL-06 10.44.42 AM

The TIMESTAMP data type stores year, month, day, hour, minute, and second, and fractional second values. When you specify TIMESTAMP as a literal, the fractional seconds precision value can be any number of digits up to 9, as follows:

TIMESTAMP '1997-01-31 09:26:50.124'

**TIMESTAMP\_TZ Data Type**

TIMESTAMP\_TZ corresponds to the SQL TIMESTAMP WITH TIMEZONE data type. It is a variant of TIMESTAMP that includes a time zone offset in its value. The time zone offset is the difference (in hours and minutes) between local time and UTC (Coordinated Universal Time—formerly Greenwich Mean Time). This data type is useful for collecting and evaluating date information across geographic regions.

Oracle time zone data is derived from the public domain information available at <ftp://elsie.nci.nih.gov/pub/>. Oracle time zone data may not reflect the most recent data available at this site.

The TIMESTAMP\_TZ data type is a variant of TIMESTAMP that includes a time zone offset. When you specify TIMESTAMP\_TZ as a literal, the fractional seconds precision value can be any number of digits up to 9. For example:

TIMESTAMP '1997-01-31 09:26:56.66 +02:00'

Two TIMESTAMP\_TZ values are considered identical if they represent the same instant in UTC, regardless of the TIME ZONE offsets stored in the data. For example,

TIMESTAMP '1999-04-15 8:00:00 -8:00'

is the same as

TIMESTAMP '1999-04-15 11:00:00 -5:00'

That is, 8:00 a.m. Pacific Standard Time is the same as 11:00 a.m. Eastern Standard Time.

You can replace the UTC offset with the TZR (time zone region) format element. For example, the following example has the same value as the preceding example:

TIMESTAMP '1999-04-15 8:00:00 US/Pacific'

To eliminate the ambiguity of boundary cases when the daylight saving time switches, use both the TZR and a corresponding TZD format element. The following example ensures that the preceding example will return a daylight saving time value:

TIMESTAMP '1999-10-29 01:30:00 US/Pacific PDT'

You can also express the time zone offset using a datetime expression.

See Also:

["Datetime and Interval Expressions"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression003.htm#BABFFDGC)

If you do not add the TZD format element, and the datetime value is ambiguous, then Oracle returns an error if you have the ERROR\_ON\_OVERLAP\_TIME session parameter set to TRUE. If that parameter is set to FALSE, then Oracle interprets the ambiguous datetime as standard time in the specified region.

**TIMESTAMP\_LTZ Data Type**

TIMESTAMP\_LTZ corresponds to the SQL TIMESTAMP WITH LOCAL TIMEZONE data type. It is another variant of TIMESTAMP that includes a time zone offset in its value. It differs from TIMESTAMP\_LTZ in that data stored in the Database is normalized to the database time zone, and the time zone offset is not stored as part of the column data. When a user retrieves the data, Oracle returns it in the user's local session time zone. The time zone offset is the difference (in hours and minutes) between local time and UTC (Coordinated Universal Time—formerly Greenwich Mean Time). This data type is useful for displaying date information in the time zone of the client system in a two-tier application.

Oracle time zone data is derived from the public domain information available at <ftp://elsie.nci.nih.gov/pub/>. Oracle time zone data may not reflect the most recent data available at this site.

The TIMESTAMP\_LTZ data type differs from TIMESTAMP\_TZ in that data stored in the Database is normalized to the database time zone. The time zone offset is not stored as part of the column data. There is no literal for TIMESTAMP\_LTZ. Rather, you represent values of this data type using any of the other valid datetime literals. The table that follows shows some of the formats you can use to add a TIMESTAMP\_LTZ value into object, along with the corresponding value returned by a OLAP DML statement such as a [SHOW](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_commands_2049.htm#i79246) command.

| **Value Specified When Adding Value** | **Value Returned** |
| --- | --- |
| '19-FEB-2004' | 19-FEB-2004.00.00.000000 AM |
| SYSTIMESTAMP | 19-FEB-04 02.54.36.497659 PM |
| TO\_TIMESTAMP('19-FEB-2004', 'DD-MON-YYYY')); | 19-FEB-04 12.00.00.000000 AM |
| SYSDATE | 19-FEB-04 02.55.29.000000 PM |
| TO\_DATE('19-FEB-2004', 'DD-MON-YYYY')); | 19-FEB-04 12.00.00.000000 AM |
| TIMESTAMP'2004-02-19 8:00:00 US/Pacific'); | 19-FEB-04 08.00.00.000000 AM |

Notice that if the value specified does not include a time component (either explicitly or implicitly, then the value returned defaults to midnight.

**YMINTERVAL Data Type**

YMINTERVAL corresponds to the SQL INTERVAL YEAR TO MONTH data type. It stores a period of time using the YEAR and MONTH datetime fields. This data type is useful for representing the difference between two datetime values when only the year and month values are significant.

Specify YMINTERVAL interval literals using the following syntax.

INTERVAL 'integer [- integer ]' YEAR|MONTH [(precision) ] [TO YEAR | MONTH ]

where

* 'integer [-integer]' specifies integer values for the leading and optional trailing field of the literal. If the leading field is YEAR and the trailing field is MONTH, then the range of integer values for the month field is 0 to 11.
* precision is the maximum number of digits in the leading field. The valid range of the leading field precision is 0 to 9 and its default value is 2.

If you specify a trailing field, it must be less significant than the leading field. For example, INTERVAL '0-1' MONTH TO YEAR is not valid.

The following YMINTERVAL literal indicates an interval of 123 years, 2 months:

INTERVAL '123-2' YEAR(3) TO MONTH

Examples of the other forms of the literal follow, including some abbreviated versions:

| **Form of Interval Literal** | **Interpretation** |
| --- | --- |
| INTERVAL '123-2' YEAR(3) TO MONTH | An interval of 123 years, 2 months. You must specify the leading field precision if it is greater than the default of 2 digits. |
| INTERVAL '123' YEAR(3) | An interval of 123 years 0 months. |
| INTERVAL '300' MONTH(3) | An interval of 300 months. |
| INTERVAL '4' YEAR | Maps to INTERVAL '4-0' YEAR TO MONTH and indicates 4 years. |
| INTERVAL '50' MONTH | Maps to INTERVAL '4-2' YEAR TO MONTH and indicates 50 months or 4 years 2 months. |
| INTERVAL '123' YEAR | Returns an error, because the default precision is 2, and '123' has 3 digits. |

You can add or subtract one INTERVAL YEAR TO MONTH literal to or from another to yield another INTERVAL YEAR TO MONTH literal. For example:

INTERVAL '5-3' YEAR TO MONTH + INTERVAL'20' MONTH =

INTERVAL '6-11' YEAR TO MONTH

**DSINTERVAL Data Type**

DSINTERVAL corresponds to the SQL INTERVAL DAY TO SECOND data type. It stores a period of time in terms of days, hours, minutes, and seconds. This data type is useful for representing the precise difference between two datetime values.

Specify DSINTERVAL interval literals using the following syntax.

INTERVAL 'integer|integer time\_expr|time\_expr

DAY|HOUR|MINUTE [(leading\_precision)] | SECOND [leading\_precision[, fractional\_seconds\_precision ])]

[ TO DAY|HOUR|MINUTE|SECOND [(fractional\_seconds\_precision) ]]

where

* integer specifies the number of days. If this value contains more digits than the number specified by the leading precision, then Oracle returns an error.
* time\_expr specifies a time in the format HH[:MI[:SS[.n]]] or MI[:SS[.n]] or SS[.n], where n specifies the fractional part of a second. If n contains more digits than the number specified by fractional\_seconds\_precision, then n is rounded to the number of digits specified by the fractional\_seconds\_precision value. You can specify time\_expr following an integer and a space only if the leading field is DAY.
* leading\_precision is the number of digits in the leading field. Accepted values are 0 to 9. The default is 2.
* fractional\_seconds\_precision is the number of digits in the fractional part of the SECOND datetime field. Accepted values are 1 to 9. The default is 6.

If you specify a trailing field, it must be less significant than the leading field. For example, INTERVAL MINUTE TO DAY is not valid. As a result of this restriction, if SECOND is the leading field, the interval literal cannot have any trailing field.

The valid range of values for the trailing field are as follows:

* HOUR: 0 to 23
* MINUTE: 0 to 59
* SECOND: 0 to 59.999999999

Examples of the various forms of DSINTERVAL literals follow, including some abbreviated versions:

| **Form of Interval Literal** | **Interpretation** |
| --- | --- |
| INTERVAL '4 5:12:10.222' DAY TO SECOND(3) | 4 days, 5 hours, 12 minutes, 10 seconds, and 222 thousandths of a second. |
| INTERVAL '4 5:12' DAY TO MINUTE | 4 days, 5 hours and 12 minutes. |
| INTERVAL '400 5' DAY(3) TO HOUR | 400 days 5 hours. |
| INTERVAL '400' DAY(3) | 400 days. |
| INTERVAL '11:12:10.2222222' HOUR TO SECOND(7) | 11 hours, 12 minutes, and 10.2222222 seconds. |
| INTERVAL '11:20' HOUR TO MINUTE | 11 hours and 20 minutes. |
| INTERVAL '10' HOUR | 10 hours. |
| INTERVAL '10:22' MINUTE TO SECOND | 10 minutes 22 seconds. |
| INTERVAL '10' MINUTE | 10 minutes. |
| INTERVAL '4' DAY | 4 days. |
| INTERVAL '25' HOUR | 25 hours. |
| INTERVAL '40' MINUTE | 40 minutes. |
| INTERVAL '120' HOUR(3) | 120 hours. |
| INTERVAL '30.12345' SECOND(2,4) | 30.1235 seconds. The fractional second '12345' is rounded to '1235' because the precision is 4. |

You can add or subtract one DAY TO SECOND interval literal from another DAY TO SECOND literal. For example.

INTERVAL'20' DAY - INTERVAL'240' HOUR = INTERVAL'10-0' DAY TO SECOND

**Boolean Data Type**

A BOOLEAN data type enables you to represent logical values. In code, BOOLEAN values are represented by values for "no" and "yes" (in any combination of uppercase and lowercase characters). The actual values that are recognized in your version of Oracle OLAP are determined by the language identified by the NLS\_LANGUAGE option. You can use the read-only NOSPELL and YESSPELL options to obtain the values represent BOOLEAN values. In English language code, you can represent BOOLEAN values, using:

* YES, TRUE, ON
* NO, FALSE, OFF

Working with BOOLEAN expressions is discussed in ["Boolean Expressions"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression003.htm#i1010516).

**RAW Data Type**

The RAW data type stores data that is not to be interpreted (that is, not explicitly converted when moving data between different systems) by Oracle Database. The RAW data type is intended for binary data or byte strings.

The syntax for specifying RAW data is as follows:

RAW (size)

where you must specify a size up to the maximum of 2000 bytes

RAW is a variable-length data type, however Oracle Net (which connects user sessions to the instance) and Oracle Database utilities do not perform character conversion when transmitting RAW data.

When Oracle automatically converts RAW data to and from text data, the binary data is represented in hexadecimal form, with one hexadecimal character representing every four bits of RAW data. For example, one byte of RAW data with bits 11001011 is displayed and entered as CB.

**Row Identifier Data Types**

The row identifier data types are used to store an address of a row in a relational table. The OLAP DML supports two different data types that you can use to copy this data from a relational table into objects in an analytic workspace:

* [ROWID Data Type](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#BABJFEAJ)
* [UROWID Data Type](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#BABDDDCI)

**ROWID Data Type**

You can examine a row address of a relational table by querying the pseudocolumn ROWID. Values of this pseudocolumn are strings representing the address of each row. These strings have the data type ROWID.

Note:

Although you can create relational tables and clusters that contain actual columns having the ROWID data type. Oracle Database does not guarantee that the values of such columns are valid rowids

The extended ROWID data type stored in a user column includes the data in the restricted rowid plus a data object number. The data object number is an identification number assigned to every database segment. You can retrieve the data object number from the data dictionary views USER\_OBJECTS, DBA\_OBJECTS, and ALL\_OBJECTS. Objects that share the same segment (clustered tables in the same cluster, for example) have the same object number.

Extended rowids are stored as base 64 values that can contain the characters A-Z, a-z, 0-9, and the plus sign (+) and forward slash (/). Extended rowids are not available directly. You can use a supplied package, DBMS\_ROWID, to interpret extended rowid contents. The package functions extract and provide information that would be available directly from a restricted rowid as well as information specific to extended rowids.

See also:

For more information on the ROWID data type and pseudocolumns, see the discussions of those topics in Oracle Database SQL Language Reference.

**UROWID Data Type**

The rows of some relational tables have addresses that are not physical or permanent or were not generated by Oracle Database. For example, the row addresses of index-organized tables are stored in index leaves, which can move. Rowids of foreign tables (such as DB2 tables accessed through a gateway) are not standard Oracle rowids.

Oracle uses universal rowids (urowids) to store the addresses of index-organized and foreign tables. Index-organized tables have logical urowids and foreign tables have foreign urowids. Both types of urowid are stored in the ROWID pseudocolumn (as are the physical rowids of heap-organized tables).

Oracle creates logical rowids based on the primary key of the table. The logical rowids do not change as long as the primary key does not change. The ROWID pseudocolumn of an index-organized table has a data type of UROWID. You can access this pseudocolumn as you would the ROWID pseudocolumn of a heap-organized table (that is, using a SELECT ... ROWID statement). If you want to store the rowids of an index-organized table, then you can define a column of type UROWID for the table and retrieve the value of the ROWID pseudocolumn into that column.

See also:

For more information on the UROWID data type, see the discussions of that data type in Oracle Database SQL Language Reference.

**Converting from One Data Type to Another**

In may cases, Oracle OLAP performs automatic data type conversion for you as discussed in ["Automatic Conversion of Textual Data Types"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJAHDFIC) and ["Automatic Conversion of Numeric Data Types"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#CJACHBEA). Additionally there area number of OLAP DML functions that you can use to convert values from one data type to another.

**Automatic Conversion of Textual Data Types**

Oracle OLAP automatically converts NTEXT values to TEXT when they are specified as arguments to OLAP DML statements. This can result in data loss when the NTEXT values cannot be represented in the database character set

**Automatic Conversion of Numeric Data Types**

Oracle OLAP automatically converts SHORTINTEGER variables, as well as INTEGER variables with a fixed width of 1 byte, to INTEGER (with a width of 4 bytes) for calculations. When you calculate a total of SHORTINTEGER variables, then you can obtain and report a result greater than 32,767 or less than -32,768. When you calculate a total of 1-byte INTEGER variables, then you can obtain and report a result greater than 127 or less than -128. However, when you try to assign the result to a SHORTINTEGER variable or a 1-byte INTEGER variable respectively, then the variable is set to NA.

Oracle OLAP automatically converts numeric data types according to the following rules:

* When you use a value with the SHORTINTEGER or SHORTDECIMAL data type in an expression, then the value is converted to its long counterpart before using it. See ["Boolean Expressions"](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression003.htm#i1010516) for information about problems that can occur when you mix SHORTDECIMAL and DECIMAL data types in a comparison expression.
* When you save the results of a calculation as a value with the SHORTINTEGER data type, then NA is stored when the result is outside the range of a SHORTINTEGER (-32768 to 32767).
* When you assign the value of a DECIMAL expression to an object with the INTEGER data type, then the value is rounded before storing or using it.

Note:

When a DECIMAL value is outside the range of an INTEGER, then an NA is stored.

* When you use a decimal value where a value with the INTEGER data type is required, then the value is rounded before storing or using it.

Note:

When the DECIMAL value is outside the range of an INTEGER, then an NA is stored.

* When you assign the value of a decimal expression to a variable with the SHORTDECIMAL data type, then only the first 7 significant digits are stored.
* When you combine NUMBER values with other numeric data types, then all values are converted to NUMBER.

When these conversion are not what you want, then you can use the CONVERT, TO\_CHAR, TO\_NCHAR, TO\_NUMBER, or TO\_DATE functions to get different results.

Top of Form

Reader Comment

subject 

from     Anonymous (or [Sign In](http://www.oracle.com/webapps/redirect/signon?nexturl=http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression001.htm#i1020862))



Comments, corrections, and suggestions are forwarded to authors every week. By submitting, you confirm you agree to the [terms and conditions](http://www.oracle.com/pls/db111/comment_disclaimer). Use the [OTN forums](http://forums.oracle.com/) for product questions. For support or consulting, file a service request through [My Oracle Support](http://myoraclesupport.oracle.com/).

Bottom of Form

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | [[Go to previous page](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression.htm) Previous](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression.htm) | [[Go to next page](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression002.htm) Next](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/dml_expression002.htm) | | Oracle Copyright © 1994, 2008, Oracle. All rights reserved. [Legal Notices](http://docs.oracle.com/cd/B28359_01/dcommon/html/cpyr.htm) | |  |  |  | | --- | --- | --- | | [[Go to Documentation Home](http://www.oracle.com/pls/db111/homepage/) Home](http://www.oracle.com/pls/db111/homepage/) | [[Go to Book List](http://www.oracle.com/pls/db111/portal.all_books) Book List](http://www.oracle.com/pls/db111/portal.all_books) | [[Go to Table of Contents](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/toc.htm) Con](http://docs.oracle.com/cd/B28359_01/olap.111/b28126/toc.htm) | |

# Term: [TIMESTAMP](http://psoug.org/definition/TIMESTAMP.htm)

Definition:  
When storing date and time information in Oracle, there are two basic options for the column's datatype: the DATE datatype and the TIMESTAMP datatype.  
  
The DATE datatype has the ability to store the month, day, year, century, hours, minutes, and seconds. It has a resolution or granularity of 1 second. To distinguish or record events with a separation of less than 1 second, the TIMESTAMP datatype is required.  
  
The TIMESTAMP datatype stores all of the information that the DATE datatype stores, but also includes fractional seconds.

Examples

Example 8-144 Converting Text Data to TIMESTAMP\_TZ DAta

SHOW TO\_TIMESTAMP\_TZ ('2006-03-26 7:33:00 -4:00', 'YYYY-MM-DD HH:MI:SS TZH:TZM')

26-MAR-06 07.33.00 AM -04:00