

**Predefined PL/SQL Scalar Data Types and Subtypes**



**Predefined PL/SQL Numeric Data Types and Subtypes**



**PLS\_INTEGER and BINARY\_INTEGER Data Types**



**NUMBER Data Type**



**Predefined PL/SQL Character Data Types and Subtypes**





**RAW Data Type**

The RAW data type stores binary or byte strings, such as sequences of graphics

characters or digitized pictures. Raw data is like VARCHAR2 data, except that PL/SQL

does not interpret raw data.

**NCHAR and NVARCHAR2 Data Types**



**LONG and LONG RAW Data Types**

The LONG data type stores variable-length character strings. The LONG data type is like

the VARCHAR2 data type, except that the maximum size of a LONG value is 32,760 bytes

(as opposed to 32,767 bytes).

The LONG RAW data type stores binary or byte strings. LONG RAW data is like LONG data,

except that LONG RAW data is not interpreted by PL/SQL. The maximum size of a LONG

RAW value is 32,760 bytes.

**ROWID and UROWID Data Types**

Internally, every database table has a ROWID pseudocolumn, which stores binary

values called rowids. Each **rowid** represents the storage address of a row. A **physical**

**rowid** identifies a row in an ordinary table.

A **logical rowid** identifies a row in an index-organized table. The ROWID data type can store only physical rowids, while the UROWID (**universal rowid**) data type can store physical, logical, or foreign (not

database) rowids.

**Predefined PL/SQL BOOLEAN Data Type**

The BOOLEAN data type stores logical values, which you can use in logical operations.

The logical values are the Boolean values TRUE and FALSE and the value NULL.

The syntax for specifying an BOOLEAN data item is:

BOOLEAN

**Predefined PL/SQL Datetime and Interval Data Types**



**Predefined PL/SQL Large Object (LOB) Data Types**

****



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Datatype** | **Description** | **Max Size: Oracle 8** | **Max Size: Oracle 9i/10g** | **Max Size: Oracle 11g** | **Max Size:  PL/SQL** | **PL/SQL Subtypes/  Synonyms** |
| VARCHAR2(*size*) | Variable length character string having maximum length *size*bytes. You must specify size | **4000**bytes minimum is 1 | **4000**bytes minimum is 1 | **4000**bytes minimum is 1 | 32,767 bytes minimum is 1 | STRING VARCHAR |
| NVARCHAR2(*size*) | Variable length national character set string having maximum length *size* bytes. You must specify size | 4000 bytes minimum is 1 | 4000 bytes minimum is 1 | 4000 bytes minimum is 1 | 32,767 bytes minimum is 1 | STRING VARCHAR |
| VARCHAR | Now deprecated (provided for backward compatibility only) VARCHAR is a synonym for VARCHAR2 but this usage might change in future versions. | - | - | - |  |  |
| CHAR(*size*) | Fixed length character data of length size bytes. This should be used for fixed length data. Such as codes A100, B102… | **2000** bytes Default and minimum size is 1 byte. | **2000** bytes Default and minimum size is 1 byte. | **2000** bytes Default and minimum size is 1 byte. | 32,767 bytes Default and minimum size is 1 byte. | CHARACTER |
| NCHAR(*size*) | Fixed length national character set data of length size bytes. This should be used for fixed length data. Such as codes A100, B102… | 2000 bytes Default and minimum size is 1 byte. | 2000 bytes Default and minimum size is 1 byte. | 2000 bytes Default and minimum size is 1 byte. | 32767 bytes Default and minimum size is 1 byte. |  |
| NUMBER(*p,s*) | Number having [precision](http://ss64.com/ora/syntax-datatypes.html#precision) *p* and scale *s*. | The precision p can range from 1 to 38.  The scale s can range from -84 to 127. | The precision p can range from 1 to 38.  The scale s can range from -84 to 127. | The precision p can range from 1 to 38.  The scale s can range from -84 to 127. | Magnitude  1E-130 .. 10E125  maximum precision of 126 binary digits, which is roughly equivalent to 38 decimal digits  The scale s can range from -84 to 127.   For floating point don't specify p,s  REAL has a maximum precision of 63 binary digits, which is roughly equivalent to 18 decimal digits | Fixed-point numbers: DEC  DECIMAL  NUMERIC   Floating-Point:  DOUBLE PRECISION FLOAT  integers: INTEGER INT  SMALLINT  BOOLEAN REAL |
| BINARY\_FLOAT | A 32-bit, single-precision floating-point number data type. Each BINARY\_FLOAT value requires 4 bytes. Supports the special values infinity and NaN (not a number). | - | Magnitude:  1.17549E-38F … 3.40282E+38F | Magnitude:  1.17549E-38F … 3.40282E+38F |  |  |
| BINARY\_DOUBLE | A 64-bit, double-precision floating-point number data type. Each BINARY\_DOUBLE value requires 8 bytes. Supports the special values infinity and NaN (not a number). | - | Magnitude: 2.22507485850720E-308  … 1.79769313486231E+308 | Magnitude: 2.22507485850720E-308  … 1.79769313486231E+308 |  |  |
| BOOLEAN | True, False or NULL | n/a Use either Number or CHAR | n/a Use either Number or CHAR | n/a Use either Number or CHAR | BOOLEAN |  |
| PLS\_INTEGER | signed integers PLS\_INTEGER values require less storage and provide better performance than NUMBER values. | PL/SQL only | PL/SQL only | PL/SQL only | magnitude range is -2,147,483,647 .. 2,147,483,647 | SIMPLE\_INTEGER a sub-type of PLS\_INTEGER that is always NOT NULL. (11g) |
| BINARY\_INTEGER | signed integers (older slower version of PLS\_INTEGER) |  |  |  | magnitude range is -2,147,483,647 .. 2,147,483,647 | NATURAL NATURALN POSITIVE POSITIVEN SIGNTYPE |
| LONG | Character data of variable length (A bigger version the VARCHAR2 datatype) | 2 Gigabytes | 2 Gigabytes - but now deprecated (provided for backward compatibility only). | 2 Gigabytes - but now deprecated (provided for backward compatibility only). | 32760 bytes Note this is smalller than the maximum width of a LONG column |  |
| DATE | Valid date range | from January 1, 4712 BC to December 31, **9999**AD. | from January 1, 4712 BC to December 31, **9999** AD. | from January 1, 4712 BC to December 31, **9999** AD. | from January 1, 4712 BC to December 31,**9999** AD.  (in Oracle7 = 4712 AD) |  |
| TIMESTAMP (*fractional\_seconds\_precision*) | the number of digits in the fractional part of the SECOND datetime field. | - | Accepted values of*fractional\_seconds\_precision*are 0 to 9. (default = 6) | Accepted values of*fractional\_seconds\_precision*are 0 to 9. (default = 6) |  |  |
| TIMESTAMP (*fractional\_seconds\_precision*) WITH {LOCAL} TIMEZONE | As above with time zone displacement value | - | Accepted values of*fractional\_seconds\_precision*are 0 to 9. (default = 6) | Accepted values of*fractional\_seconds\_precision*are 0 to 9. (default = 6) |  |  |
| INTERVAL YEAR (*year\_precision*) TO MONTH | Time in years and months, where *year\_precision* is the number of digits in the YEAR datetime field. | - | Accepted values are 0 to 9. (default = 2) | Accepted values are 0 to 9. (default = 2) |  |  |
| INTERVAL DAY (*day\_precision*) TO SECOND (*fractional\_seconds\_precision*) | Time in days, hours, minutes, and seconds.  *day\_precision* is the maximum number of digits in 'DAY'   *fractional\_seconds\_precision* is the max number of fractional digits in the SECOND field. | - | *day\_precision* can be 0 to 9. (default = 2)  *fractional\_seconds\_precision*can be 0 to 9. (default = 6) | *day\_precision* can be 0 to 9. (default = 2)  *fractional\_seconds\_precision*can be 0 to 9. (default = 6) |  |  |
| RAW(*size*) | Raw binary data of length size bytes. You must specify size for a RAW value. | Maximum size is**2000** bytes | Maximum size is **2000** bytes | Maximum size is **2000** bytes | 32767 bytes |  |
| LONG RAW | Raw binary data of variable length. (not intrepreted by PL/SQL) | 2 Gigabytes. | 2 Gigabytes - but now deprecated (provided for backward compatibility only) | 2 Gigabytes - but now deprecated (provided for backward compatibility only) | 32760 bytes Note this is smalller than the maximum width of a LONG RAW column |  |
| [ROWID](http://www.orafaq.com/wiki/ROWID) | Hexadecimal string representing the unique address of a row in its table. (primarily for values returned by the ROWID pseudocolumn.) | 10 bytes | 10 bytes | 10 bytes | Hexadecimal string representing the unique address of a row in its table. (primarily for values returned by the ROWID pseudocolumn.) |  |
| UROWID | Hex string representing the logical address of a row of an index-organized table | The maximum size and default is 4000 bytes | The maximum size and default is 4000 bytes | The maximum size and default is 4000 bytes | universal rowid - Hex string representing the logical address of a row of an index-organized table, either physical, logical, or foreign (non-Oracle) | See[CHARTOROWID](http://ss64.com/ora/syntax-functions.html)and the package:[DBMS\_ROWID](http://ss64.com/orap/DBMS_ROWID.html) |
| MLSLABEL | Binary format of an operating system label.This datatype is used with Trusted Oracle7. |  |  |  |  |  |
| CLOB | Character Large Object | 4 Gigabytes | 8 TB | 8 TB to 128 TB  (4 Gigabytes - 1) \* (database block size) |  |  |
| NCLOB | National Character Large Object | 4 Gigabytes | 8 TB | 8 TB to 128 TB  (4 Gigabytes - 1) \* (database block size) |  |  |
| BLOB | Binary Large Object | 4 Gigabytes | 8 TB | 8 TB to 128 TB  (4 Gigabytes - 1) \* (database block size) |  |  |
| BFILE | pointer to binary file on disk | 4 Gigabytes | 8 TB | 8 TB to 128 TB  (4 Gigabytes - 1) \* (database block size) |  |  |
| XMLType | A system-defined type for storing binary XML data stored internally as a CLOB (starting with 11gR1 stored as a BLOB) | - | 64K | In 11gR1 the limit is 2G/4G depending on DB character set. | Populate with XML from a CLOB, BLOB or VARCHAR2.  or query from another XMLType column. |  |

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Valid Datetime Values** | **Valid Interval Values** |
| YEAR | -4712 to 9999 (excluding year 0) | Any nonzero 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 calendar for the locale) | Any nonzero 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 DATE. | 0 to 59.9(n), where 9(n) is the precision of interval fractional seconds |
| TIMEZONE\_HOUR | -12 to 14 (range accommodates daylight savings time changes)  Not applicable for DATE or TIMESTAMP. | Not applicable |
|  |  |  |
| TIMEZONE\_MINUTE | 00 to 59  Not applicable for DATE or TIMESTAMP. | Not applicable |
| TIMEZONE\_REGION | Not applicable for DATE or TIMESTAMP. | Not applicable |
| TIMEZONE\_ABBR | Not applicable for DATE or TIMESTAMP. | Not applicable |

|  |  |
| --- | --- |
| **S.No** | **Function Name & Description** |
| 1 | **ADD\_MONTHS(x, y);**  Adds **y** months to **x**. |
| 2 | **LAST\_DAY(x);**  Returns the last day of the month. |
| 3 | **MONTHS\_BETWEEN(x, y);**  Returns the number of months between **x** and **y**. |
| 4 | **NEXT\_DAY(x, day);**  Returns the datetime of the next *day* after **x**. |
| 5 | **NEW\_TIME;**  Returns the time/day value from a time zone specified by the user. |
| 6 | **ROUND(x [, unit]);**  Rounds **x**. |
| 7 | **SYSDATE();**  Returns the current datetime. |
| 8 | **TRUNC(x [, unit]);**  Truncates **x**. |