Oracle Datatypes

1.Data types for Oracle 7- Oracle 11g + PL/SQL

Datatype	**Description**	**Max Size: Oracle 7**	**Max Size: Oracle 8**	**Max Size: Oracle 9i/10g/11g**	**Max Size: PL/SQL**	**PL/SQL Subtypes/ Synonyms**
VARCHAR2(size)	Variable length character string having maximum length *size* bytes. You must specify size	2000 bytes minimum is 1	**4000** bytes minimum is 1	**4000** bytes minimum is 1	32767 bytes minimum is 1	STRING VARCHAR
NVARCHAR2(size)	Variable length national character set string having maximum length *size* bytes. You must specify size	N/A	4000 bytes minimum is 1	4000 bytes minimum is 1	32767 bytes minimum is 1	STRING VARCHAR
VARCHAR	Now deprecated (provided for backward compatibility only) VARCHAR is a synonym for VARCHAR2 but this usage may 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...	255 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.	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...	N/A	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 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 binary_float (32 bit) binary_double (64 bit) integers: INTEGER INT SMALLINT simple_integer(10g) BOOLEAN REAL
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 -2147483647 .. 2147483647	

	So use PLS_INTEGER where you can!					
BINARY_INTEGER	signed integers (older slower version of PLS_INTEGER)				magnitude range is -2147483647 .. 2147483647	NATURAL NATURALN POSITIVE POSITIVEN SIGNTYPE
LONG	Character data of variable length (A bigger version the VARCHAR2 datatype)	2 Gigabytes	2 Gigabytes	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, 4712 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)		
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)		
INTERVAL YEAR (year_precision) TO MONTH	Time in years and months, where year_precision is the number of	-	-	Accepted values are 0 to 9 (default = 2)		

	digits in the YEAR datetime field.					
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* may be 0 to 9 (default = 2) *fractional_seconds_precision* may 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 255 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	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	Hexadecimal string representing the unique address of a row in its table. (primarily for values returned by the ROWID pseudocolumn.)	8 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	N/A	The maximum size and	The maximum size and	universal rowid - Hex string representing the logical address of a row	See CHARTOROWID and the

	row of an index-organized table		default is 4000 bytes	default is 4000 bytes	of an index-organized table, either physical, logical, or foreign (non-Oracle)	package: DBMS_ROWID
MLSLABEL	Binary format of an operating system label.This datatype is used with Trusted Oracle7.					
CLOB	Character Large Object	4 Gigabytes	4 Gigabytes	4 Gigabytes In Oracle 11g the Max size = (4Gigabytes-1)* database block size)	4Gigabytes	
NCLOB	National Character Large Object		4 Gigabytes	4 Gigabytes In Oracle 11g the Max size = (4Gigabytes-1)* database block size)	4Gigabytes	
BLOB	Binary Large Object		4 Gigabytes	4 Gigabytes In Oracle 11g the Max size = (4Gigabytes-1)*(database block size)	4Gigabytes	
BFILE	pointer to binary file on disk		4 Gigabytes	4 Gigabytes In Oracle 11g the Max size =	The size of a BFILE is system dependent but cannot exceed four gigabytes (2**32 - 1 bytes).	

				(4Gigabytes-1)*(database block size)		
XMLType	XML data	-	-	4 Gigabytes	Populate with XML from a CLOB or VARCHAR2. or query from another XMLType column.	

2. ***Notes and Examples***

VARCHAR2
Storing character data as Varchar2 will save space:

Store 'SMITH' not 'SMITH   '

CHAR
Over time, when varchar2 columns are updated they will sometimes create chained rows - because CHAR columns are fixed width they are not affected by this - so less DBA effort is required to maintain performance.

PL/SQL
When retrieving data for a NUMBER column, consider using the PL/SQL datatype: PLS_INTEGER for better performance.

LONG
Use BLOB instead of LONG

INTEGER
This ANSI datatype will be accepted by Oracle - it is actually a synonym for NUMBER(38)

FLOAT
This ANSI datatype will be accepted by Oracle - Very similar to NUMBER it stores zero, positive, and negative floating-point numbers

NUMBER
Stores zero, positive, and negative numbers, fixed or floating-point numbers

- Fixed-point NUMBER
  NUMBER(*p*,*s*)
  precision *p* = length of the number in digits
  scale *s* = places after the decimal point, or (for negative scale values) significant places before the decimal point.
- Integer NUMBER
  NUMBER(*p*)
  This is a fixed-point number with precision *p* and scale 0. Equivalent to NUMBER(*p*,0)
- Floating-Point NUMBER
  NUMBER
  floating-point number with decimal precision 38

Confusingly the Units of measure for PRECISION vary according to the datatype.
For NUMBER data types: precision *p* = Number of Digits
For FLOAT data types: precision *p* = Binary Precision (multiply by 0.30103 to convert)

{So FLOAT = FLOAT (126) = 126 x 0.30103 = approx 37.9 digits of precision.}

Example

The value 7,456,123.89 will display as follows

NUMBER(9)	7456124
NUMBER(9,1)	7456123.9
NUMBER(*,1)	7456123.9
NUMBER(9,2)	7456123.89
NUMBER(6)	[not accepted exceeds precision]
NUMBER(7,-2)	7456100
NUMBER	7456123.89
FLOAT	7456123.89
FLOAT(12)	7456000.0

Storing Varchar2 Data

For VARCHAR2 variable whose maximum size is **less** than 2,000 bytes (or for a CHAR variable), PL/SQL allocates enough memory for the maximum size at compile time.

For a VARCHAR2 whose maximum size is 2,000 bytes **or more**, PL/SQL allocates enough memory to store the actual value at run time. In this way, PL/SQL optimizes smaller VARCHAR2 variables for performance and larger ones for efficient memory use.

For example, if you assign the same 500-byte value to VARCHAR2(1999 BYTE) and VARCHAR2(2000 BYTE) variables, PL/SQL allocates 1999 bytes for the former variable at compile time and 500 bytes for the latter variable at run time.

Storing Numeric Data

Oracle stores all numeric data in variable length format - storage space is therefore dependent on the length of all the individual values stored in the table. Precision and scale settings do not affect storage requirements. DATA_SCALE may appear to be truncating data, but Oracle still stores the exact values as input. DATA_PRECISION can be used to constrain input values.

It is possible to save storage space by having an application truncate a fractional value before inserting into a table, but you have to be very sure the business logic makes sense.

```
Select COLUMN_NAME, DATA_TYPE, DATA_LENGTH, DATA_PRECISION, DATA_SCALE
From cols Where table_name = 'Your_Table';
```

A common space-saving trick is storing **boolean** values as an Oracle CHAR, rather than NUMBER:

Create TABLE my_demo (accountcode NUMBER, postableYN CHAR check (postableYN in (0,1)));

```
-- Standard Boolean values: False=0 and True=1
Insert into my_demo values(525, '1');
Insert into my_demo values(526, '0');

Select accountcode, decode(postableYN,1,'True',0,'False') FROM my_demo;
-- or in French:
Select accountcode, decode(postableYN,1,'Vrai',0,'Faux') FROM my_demo;
```

3. *Comparison with other RDBMS's*

	int10	**int6**	**int1**	**char(n)**	**blob**	**XML**
Oracle 11	NUMBER(10)	NUMBER(6)	NUMBER(1)	VARCHAR2(n)	BLOB	XMLType
MS SQL Server 2005	NUMERIC(10)	NUMERIC(6)	TINYINT	VARCHAR(n)	IMAGE	XML
Sybase system 10	NUMERIC(10)	NUMERIC(6)	NUMERIC(1)	VARCHAR(n)	IMAGE	
MS Access (Jet)	Long Int or Double	Single	Byte	TEXT(n)	LONGBINARY	
TERADATA	INTEGER	DECIMAL(6)	DECIMAL(1)	VARCHAR(n)	VARBYTE(20480)	
DB2	INTEGER	DECIMAL(6)	DECIMAL(1)	VARCHAR(n)	VARCHAR(255)	
RDB	INTEGER	DECIMAL(6)	DECIMAL(1)	VARCHAR(n)	LONG VARCHAR	
INFORMIX	INTEGER	DECIMAL(6)	DECIMAL(1)	VARCHAR(n)	BYTE	
RedBrick	integer	int	int	char(n)	char(1024)	
INGRES	INTEGER	INTEGER	INTEGER	VARCHAR(n)	VARCHAR(1500)	

Also consider the maximum length of a table name (or column name) and the maximum size of an SQL statement - these limits vary considerably between products and versions.