There are a greater number of local variable datatypes available in PLSQL than in SQL, this part of the section examines some datatypes which are different from the SQL datatypes and examines their usefulness.

Firstly, examining the varchar2 datatype, in SQL its maximum declarable length is 4000 bytes, however in PLSQL its 32767, in addition, it has synonyms of *string* and *varchar* (not recommended). All of these declarations are legal …

```
l_var    varchar2(1);
l_var1   varchar2(32767);
l_var2   string(4001);
```

Oracle 10g allows the Developer to use reserved words etc as identifiers for local variables, this is achieved by using double quotes around the variable, for example …

```
declare

   "begin" varchar2(1000);

begin

   "begin" := 'Hello';
   dbms_output.put_line('The value of begin is '||"begin");

end;
```

Note that the variable name now becomes 'case sensitive' …

```
 dbms_output.put_line('The value of begin is '||"Begin");
                                               *
ERROR at line 9:
PLS-00201: identifier 'Begin' must be declared
```

Double quotes will allow variables to have spaces and symbols …

```
declare
   "begin the begin $" varchar2(1000);
begin
   "begin the begin $" := 'Hello';
   dbms_output.put_line('The value of Begin is '||"begin the begin $");
end;
```

The following are examples of what will now compile in PLSQL (but are not necessarily recommended) …

```
declare

   "www.address" varchar2(1000);
   "The end" varchar2(1000);
   "declare" varchar2(1000);
   "start,end" varchar2(1000);
   "form-start" varchar2(1000);

begin
```

There are a number of numeric variables the Developer can declare in PLSQL, these are as follows, these are subtypes of number or synonyms of it, therefore the maximum size of any of them is 38 bytes …

Number

Dec

Decimal

The Number variable allows up to 38 digits to be declared with the same number of decimal places if required …

l_var number(38,38);

The size of the field is termed the *precision*, the number of decimal places (if required), the *scale*.

The scale can be between -84 and 127, in real terms this can be used to truncate values entered into the variable, for example this number is initialized with 11 but the -1 scale will remove one character in from the decimal point, in other words, the Developer can control significant numbers …

l_num number(3, -1) := 11;

This will store as 10

l_num number(4, -2) := 111;

This will store as 100

Positive scale values have the opposite effect, in that any decimal places declared larger than the overall precision will expect values with leading zeroes after the decimal place …

In this example the Developer has declared a variable four bytes with 10 decimal places, therefore only values between 0.0000000001 and 0.0000009999 will be acceptable …

l_num number(4, 10) := 0.0000009999;

Any number larger such as 0.00001000 will result in this error …

ORA-06502: PL/SQL: numeric or value error: number precision too large

Any number smaller than 0.0000000001 will store the value as a 0.

Dec, Decimal and Numeric are synonyms of Number and therefore behave the same way.

The following declarations are valid …

```
l_dec       dec(10) := 0.0000009999;
l_decimal   decimal(38);
l_numeric   numeric(10);
```

It is entirely possible to declare a scale for a number without a precision, this is achieved thus, note that the size of the variable will default to 38 and that this technique is only available in PLSQL, it cannot be used in SQL …

```
create table temp (col1 number(*, 3));
```

PLSQL also supports the declaration of Integer, Int and Smallint which are again ANSI standard datatypes however like Dec, Decimal and Numeric, Oracle will internally convert them into a Number datatype with the same behavioural attributes, for example an integer can be declared with decimal places.

```
l_integer integer(12,3) := 12.34;
l_int int := 12;
l_smallint smallint := 100;
```