Advanced Techniques - %Rowtype

%Rowtype is effectively another datatype for use within a PL/SQL object, the datatype being a complete row specification from a table. This is similar to the %type datatype introduced earlier but now encompasses the entire record !

The advantage of using %rowtype is that firstly Developers need not be too concerned with Tables being altered in the Database and secondly it can be populated more simply than using many separate variables, it is also immaterial whether it is populated with all columns or merely a selection from the cursor.

Advanced Techniques - Declaring %Rowtype

%Rowtype is declared within the Declaration section of a PL/SQL object …

```
declare

    stock_inf stock_information%rowtype;

begin
```

This will create a local variable which is the same shape as one row in the Stock_Information table.

Advanced Techniques - Referencing %Rowtype

Individual columns within the rowtype can be referenced using dot notation …

<rowtype_name>.<column_name>

```
if stock_inf.si_created_date = sysdate then
        <process>;
end if;
```

Advanced Techniques - Populating %Rowtype

%Rowtype can be populated in a variety of ways …

The simplest method is a direct assignment to a single column within the structure …

stock_inf.si_section_number := 1;

Remember that a rowtype merely maps the columns of a table, it does not inherit the mandatory columns, primary indexes, check constraints or foreign key indexes … all it has are the same columns, the same column names and the same datatypes as the columns.

Advanced Techniques - Populating %Rowtype

The most common method however is by using a cursor, either Implicit or Explicit …

Implicit (single row) …

```
select si_stock_number
      ,si_section_number
      ,si_stock_description
      ,si_unit_price
into stock_inf
from stock_information
where si_stock_number = 1
and si_section_number = 12;
```

Advanced Techniques - Populating %Rowtype

An Implicit multiple row has already been covered in the course …
effectively a *'For'* loop creates a rowtype in this case *stock_inf*…

```
for stock_inf in (select si_stock_number
                        ,si_stock_description
                        ,si_unit_price
                  from stock_information) loop
```

Advanced Techniques - Populating %Rowtype

An Explicit cursor populates a rowtype this way ...

open get_records;
fetch get_records into stock_inf;
close get_records;

A rowtype is again created when in a *'For'* loop ..

Advanced Techniques - Passing %Rowtype

It is entirely possible to pass a complete record from one process to another using rowtype, providing both processes use exactly the same declaration, the sending process in the Declaration section and the receiving in the Parameter section …

Process_one has this in its declaration section …

stock_inf stock_information%rowtype;

And calls process_two thus …

process_two(stock_inf);

Advanced Techniques - Passing %Rowtype

Process_two has the following parameter …

create or replace procedure process_two
(p_record in out stock_information%rowtype) is

And will reference the values passed to it using dot notation …

p_record.si_stock_number := 12;

By specifying *IN OUT* process_two can change the values in the row and pass them back to the calling module …

Advanced Techniques - Cursor %Rowtype

A rowtype can not only be based on a table or view but can also be based on a Cursor, this means that a more complex rowtype can be created which may be made up of more than one table …

In the following example, the section number is translated to its name, note the use of an alias for each column to make the use of rowtype more readable …

Advanced Techniques - Cursor %Rowtype

```
create or replace procedure process_one is

   cursor get_stock_details is
         select b.sd_section_name                          section_name
               ,a.si_stock_number                          stock_number
               ,a.si_stock_description                     stock_description
               ,a.si_unit_price                            unit_price
               ,a.si_no_in_stock                           no_in_stock
               ,a.si_unit_price * a.si_no_in_stock         total_value
         from stock_information a
              ,section_details b
         where a.si_section_number = b.sd_section_number;

stock_inf get_stock_details%rowtype;
```

Advanced Techniques - Cursor %Rowtype

```
begin

   open get_stock_details;
   fetch get_stock_details into stock_inf;
   close get_stock_details;

   dbms_output.put_line(to_char(stock_inf.total_value));

end;
```