PL/SQL Records

What are records?

Records are another type of datatypes which oracle allows to be defined as a placeholder. Records are composite datatypes, which means it is a combination of different scalar datatypes like char, varchar, number etc.  Each scalar data types in the record holds a value. A record can be visualized as a row of data. It can contain all the contents of a row.

Declaring a record:

To declare a record, you must first define a composite datatype; then declare a record for that type.   
  
The General Syntax to define a composite datatype is:

|  |
| --- |
| TYPE record\_type\_name IS RECORD  (first\_col\_name column\_datatype,  second\_col\_name column\_datatype, ...); |

* *record\_type\_name* – it is the name of the composite type you want to define.
* *first\_col\_name, second\_col\_name, etc.,- it is the* names the fields/columns within the record.
* *column\_datatype* defines the scalar datatype of the fields.

There are different ways you can declare the datatype of the fields.

1) You can declare the field in the same way as you declare the fieds while creating the table.   
2) If a field is based on a column from database table, you can define the field\_type as follows:

|  |
| --- |
| *col\_name* table\_name.column\_name%type; |

By declaring the field datatype in the above method, the datatype of the column is dynamically applied to the field.  This method is useful when you are altering the column specification of the table, because you do not need to change the code again.

**NOTE:** You can use also*%type* to declare variables and constants.   
  
The General Syntax to declare a record of a uer-defined datatype is:

|  |
| --- |
| record\_name record\_type\_name; |

The following code shows how to declare a record called *employee\_rec* based on a user-defined type.

|  |
| --- |
| DECLARE  TYPE employee\_type IS RECORD  (employee\_id number(5),   employee\_first\_name varchar2(25),   employee\_last\_name employee.last\_name%type,   employee\_dept employee.dept%type);   employee\_salary employee.salary%type;  employee\_rec employee\_type; |

If all the fields of a record are based on the columns of a table, we can declare the record as follows:

|  |
| --- |
| record\_name table\_name%ROWTYPE; |

For example, the above declaration of employee\_rec can as follows:

|  |
| --- |
| DECLARE   employee\_rec employee%ROWTYPE; |

The advantages of declaring the record as a ROWTYPE are:  
1)  You do not need to explicitly declare variables for all the columns in a table.   
2) If you alter the column specification in the database table, you do not need to update the code.

The disadvantage of declaring the record as a ROWTYPE is:  
1) When u create a record as a ROWTYPE, fields will be created for all the columns in the table and memory will be used to create the datatype for all the fields. So use ROWTYPE only when you are using all the columns of the table in the program.

**NOTE:** When you are creating a record, you are just creating a datatype, similar to creating a variable. You need to assign values to the record to use them.  
  
The following table consolidates the different ways in which you can define and declare a pl/sql record.

|  |  |
| --- | --- |
| **Syntax** | **Usage** |
| TYPE record\_type\_name IS RECORD (column\_name1 datatype, column\_name2 datatype, ...); | Define a composite datatype, where each field is scalar. |
| col\_name table\_name.column\_name%type; | Dynamically define the datatype of a column based on a database column. |
| record\_name record\_type\_name; | Declare a record based on a user-defined type. |
| record\_name table\_name%ROWTYPE; | Dynamically declare a record based on an entire row of a table. Each column in the table corresponds to a field in the record. |

**Passing Values To and From a Record**

When you assign values to a record, you actually assign values to the fields within it.   
The General Syntax to assign a value to a column within a record direclty is:

|  |
| --- |
| record\_name.col\_name := value; |

If you used %ROWTYPE to declare a record, you can assign values as shown:

|  |
| --- |
| record\_name.column\_name := value; |

We can assign values to records using SELECT Statements as shown:

|  |
| --- |
| SELECT col1, col2  INTO record\_name.col\_name1, record\_name.col\_name2  FROM table\_name  [WHERE clause]; |

If %ROWTYPE is used to declare a record then you can directly assign values to the whole record instead of each columns separately. In this case, you must SELECT all the columns from the table into the record as shown:

|  |
| --- |
| SELECT \* INTO record\_name  FROM table\_name  [WHERE clause]; |

Lets see how we can get values from a record.   
The General Syntax to retrieve a value from a specific field into another variable is:

|  |
| --- |
| var\_name := record\_name.col\_name; |

The following table consolidates the different ways you can assign values to and from a record:

|  |  |
| --- | --- |
| **Syntax** | **Usage** |
| record\_name.col\_name := value; | To directly assign a value to a specific column of a record. |
| record\_name.column\_name := value; | To directly assign a value to a specific column of a record, if the record is declared using %ROWTYPE. |
| SELECT col1, col2 INTO record\_name.col\_name1, record\_name.col\_name2 FROM table\_name [WHERE clause]; | To assign values to each field of a record from the database table. |
| SELECT \* INTO record\_name FROM table\_name [WHERE clause]; | To assign a value to all fields in the record from a database table. |
| variable\_name := record\_name.col\_name; | To get a value from a record column and assigning it to a variable. |