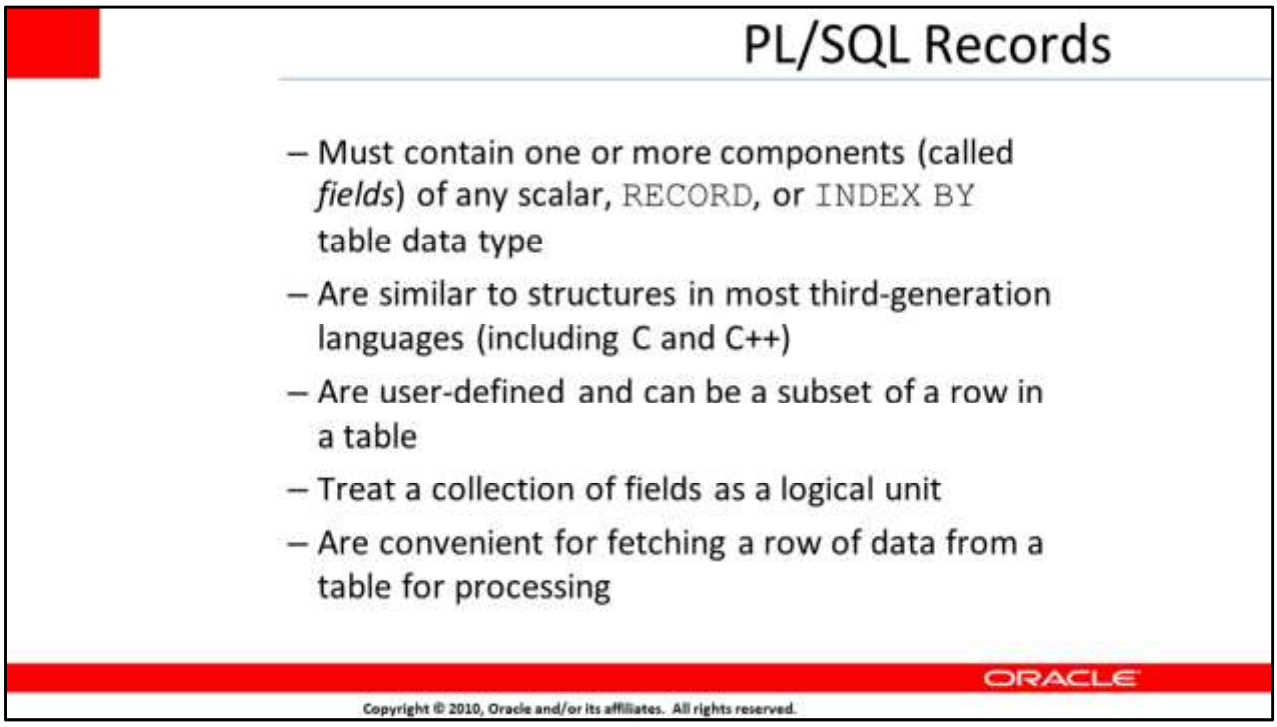


PL/SQL Records

A record is a group of related data items stored in fields, each with its own name and data type.

Each record defined can have as many fields as necessary. Records can be assigned initial values and can be defined as NOT NULL.

Fields without initial values are initialized to NULL.

The DEFAULT keyword as well as := can be used in initializing fields.

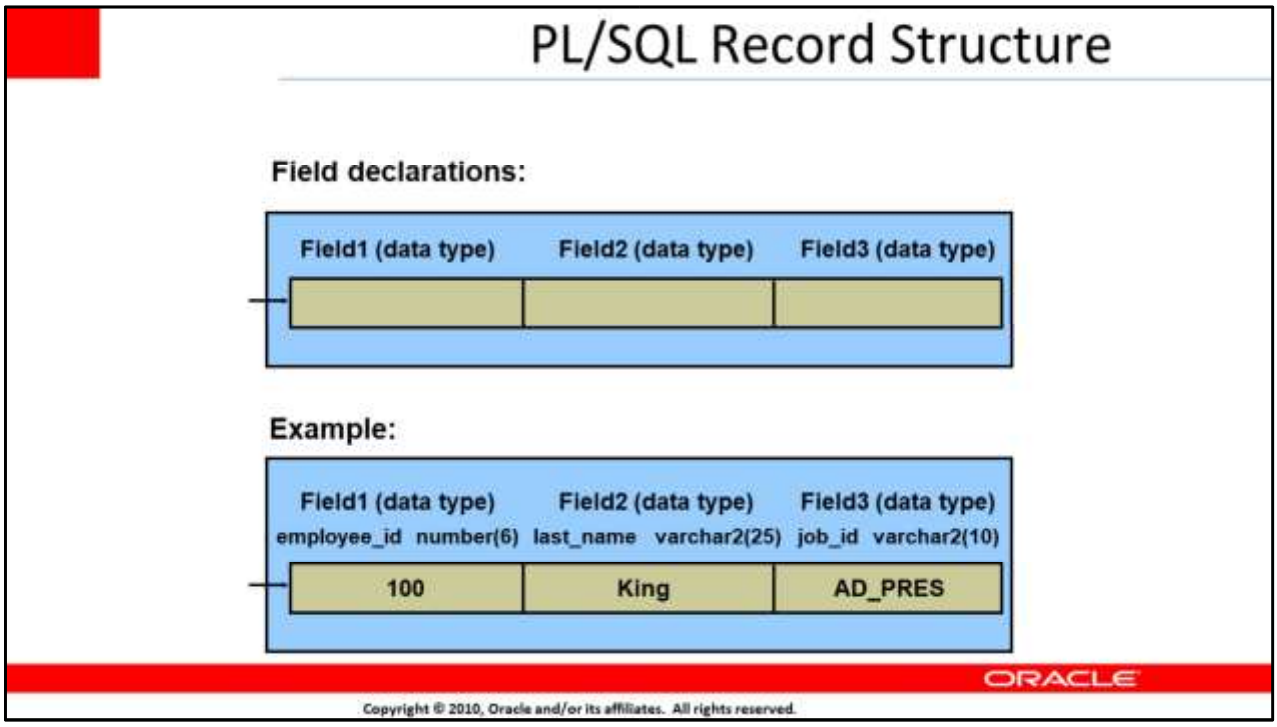
You can define RECORD types and declare user-defined records in the declarative part of any block, subprogram, or package.

You can declare and reference nested records. One record can be the component of another record.

Creating a PL/SQL Record

PL/SQL records are user-defined composite types. To use them, perform the following steps:

1. Define the record in the declarative section of a PL/SQL block. The syntax for defining the record is shown in the slide.



2. Declare (and optionally initialize) the internal components of this record type.In the syntax:

type_name	Is the name of the RECORD type (This identifier is used to declare records.)
field_name	Is the name of a field within the record
field_type	Is the data type of the field (It represents any PL/SQL data type except REFCURSOR. You can use the %TYPE and %ROWTYPE attributes.)
expr	Is the field_type or an initial value

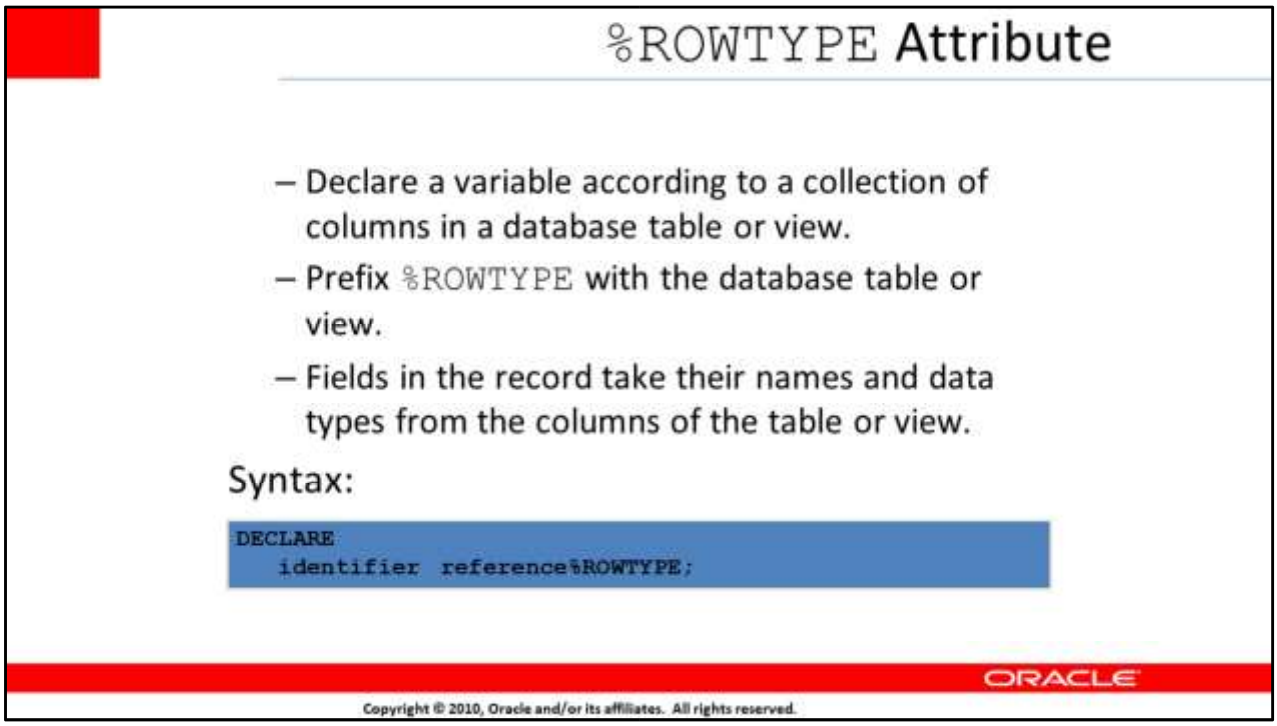
The NOTNULL constraint prevents assigning of nulls to the specified fields. Be sure to initialize the NOTNULL fields.

PL/SQL Record Structure

Fields in a record are accessed with the name of the record. To reference or initialize an individual field, use the dot notation:

record_name.field_name

For example, you reference the job_id field in the emp_record record as follows:



emp_record.job_id

You can then assign a value to the record field: emp_record.job_id

:= 'ST_CLERK';

In a block or subprogram, user-defined records are instantiated when you enter the block or subprogram. They cease to exist when you exit the block or subprogram.

%ROWTYPE Attribute

You learned that %TYPE is used to declare a variable of the column type. The variable has the same data type and size as the table column. The benefit of %TYPE is that you do not have to change the variable if the column is altered. Also, if the variable is a number and is used in any calculations, you need not worry about its precision.

The %ROWTYPE attribute is used to declare a record that can hold an entire row of a table or view. The fields in the record take their names and data types from the columns of the table or view. The record can also store an entire row of data fetched from a cursor or cursor variable.

The slide shows the syntax for declaring a record. In the syntax:

identifier	Is the name chosen for the record as a whole
reference	Is the name of the table, view, cursor, or cursor variable on which the record is to be based (The table or view must exist for this reference to be valid.)

In the following example, a record is declared using %ROWTYPE as a data type specifier:

```
DECLARE emp_record
 employees%ROWTYPE;
 ...
```

%ROWTYPE Attribute (continued)

The emp_record record has a structure consisting of the following fields, each representing a column in the employees table.

Note: This is not code, but simply the structure of the composite variable.

(employee_id NUMBER(6), first_name
VARCHAR2(20), last_name
VARCHAR2(20), email
VARCHAR2(20), phone_number
VARCHAR2(20), hire_date DATE,
salary NUMBER(8,2), commission_pct
NUMBER(2,2), manager_id
NUMBER(6), department_id NUMBER(4))

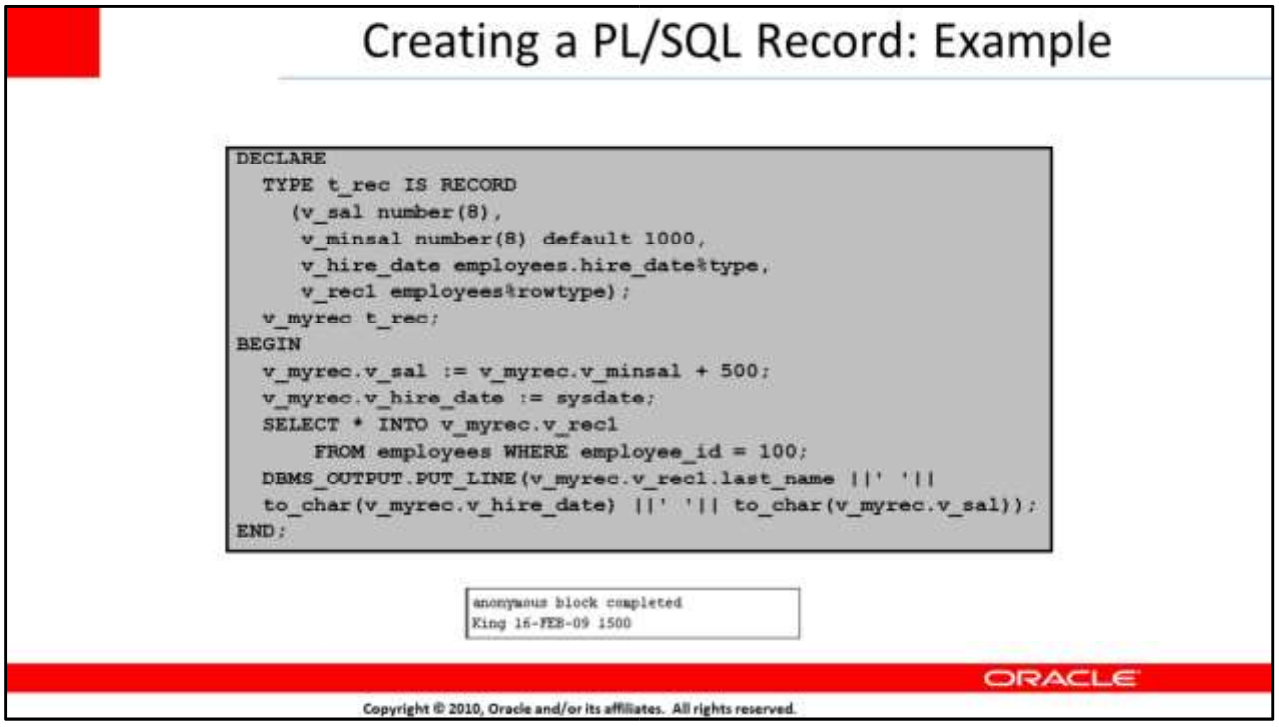
To reference an individual field, use the dot notation: record_name.field_name

For example, you reference the commission_pct field in the emp_record record as follows:

emp_record.commission_pct You can then assign a value to the record field:
emp_record.commission_pct:= .35;

Assigning Values to Records

You can assign a list of common values to a record by using the SELECT or FETCH statement. Make sure that the column names appear in the same order as the fields in your record. You can also assign one record to another if both have the same corresponding data types. A record of type employees%ROWTYPE and a userdefined record type having analogous fields of the employees table will have the same data type. Therefore, if a user-defined record contains fields similar to the fields of a %ROWTYPE record, you can assign that user-defined record to the %ROWTYPE record.



Creating a PL/SQL Record: Example

The field declarations used in defining a record are like variable declarations. Each field has a unique name and a specific data type. There are no predefined data types for PL/SQL records, as there are for scalar variables. Therefore, you must create the record type first, and then declare an identifier using that type. In the example in the slide, a PL/SQL record is created using the required twostep process:

1. A record type (t_rec) is defined
2. A record (v_myrec) of the t_rec type is declared

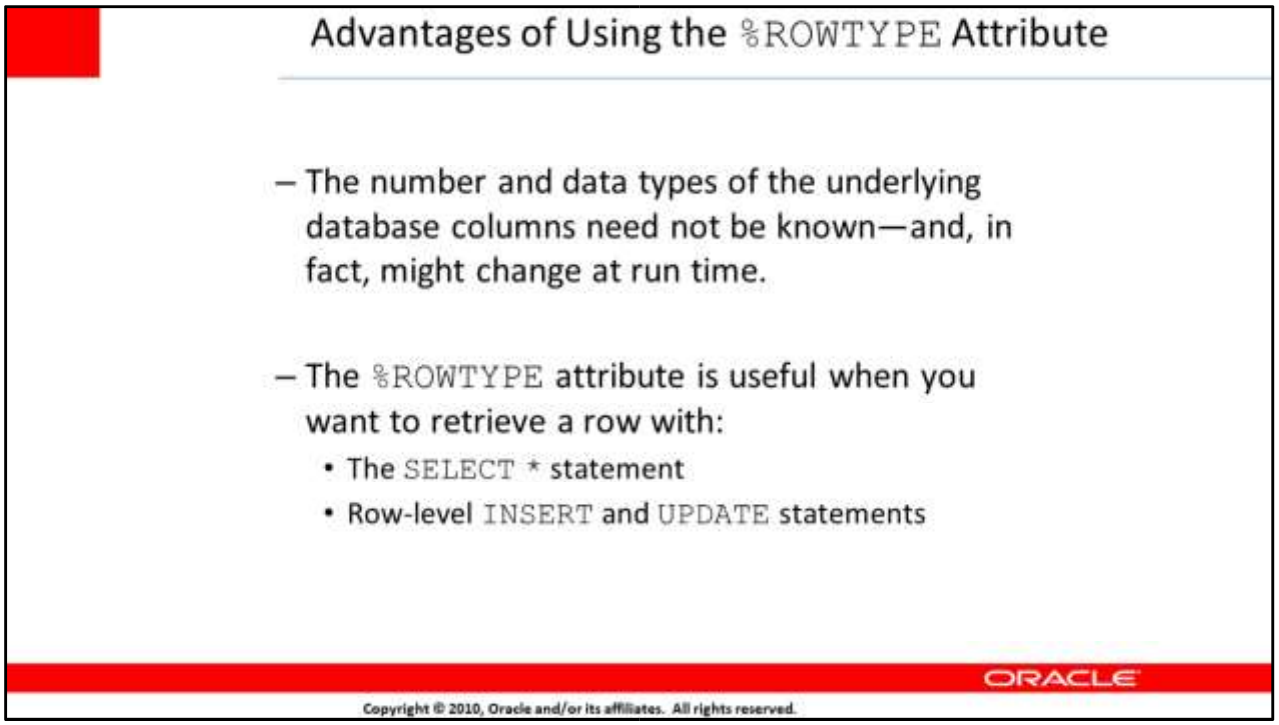
Note

The record contains four fields: v_sal, v_minsal, v_hire_date, and v_rec1.

v_rec1 is defined using the %ROWTYPE attribute, which is similar to the %TYPE attribute. With %TYPE, a field inherits the data type of a specified column. With %ROWTYPE, a field inherits the column names and data types of all columns in the referenced table.

PL/SQL record fields are referenced using the <record>.<field> notation, or the <record>.<field>.<column> notation for

fields that are defined with the %ROWTYPE attribute.
You can add the NOTNULL constraint to any field declaration to prevent assigning nulls to that field. Remember that fields that are declared as NOTNULL must be initialized.

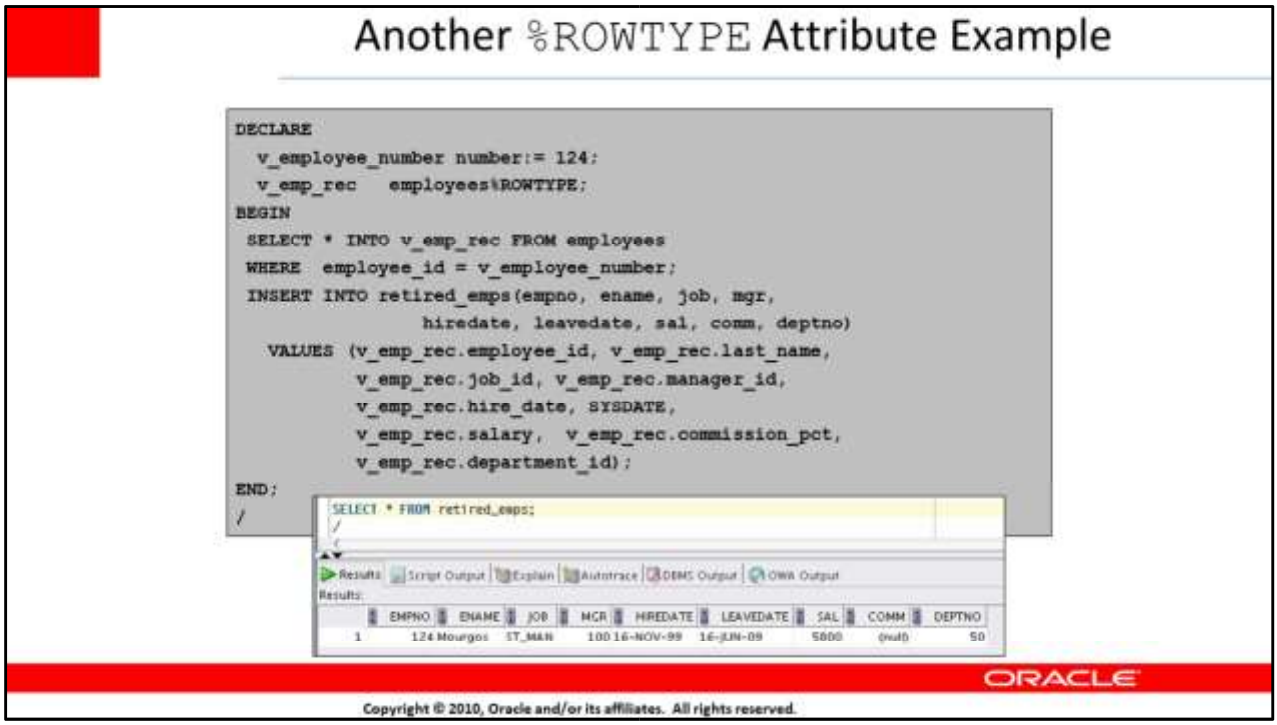


Advantages of Using %ROWTYPE

The advantages of using the %ROWTYPE attribute are listed in the slide. Use the %ROWTYPE attribute when you are not sure about the structure of the underlying database table.

The main advantage of using %ROWTYPE is that it simplifies maintenance. Using %ROWTYPE ensures that the data types of the variables declared with this attribute change dynamically when the underlying table is altered. If a DDL statement changes the columns in a table, the PL/SQL program unit is invalidated. When the program is recompiled, it automatically reflects the new table format.

The %ROWTYPE attribute is particularly useful when you want to retrieve an entire row from a table. In the absence of this attribute, you would be forced to declare a variable for each of the columns retrieved by the SELECT statement.



Another %ROWTYPE Attribute Example

Another example of the %ROWTYPE attribute is shown in the slide. If an employee is retiring, information about that employee is added to a table that holds information about retired employees. The user supplies the employee number. The record of the employee specified by the user is retrieved from the employees table and stored in the emp_rec variable, which is declared using the %ROWTYPE attribute.

The CREATE statement that creates the retired_emps table is:

```
CREATE TABLE retired_emps
(EMPNO     NUMBER(4), ENAME     VARCHAR2(10),
 JOB       VARCHAR2(9),MGR      NUMBER(4),
 HIREDATE  DATE, LEAVEDATE  DATE,
 SAL       NUMBER(7,2), COMM      NUMBER(7,2),
 DEPTNO    NUMBER(2))
```

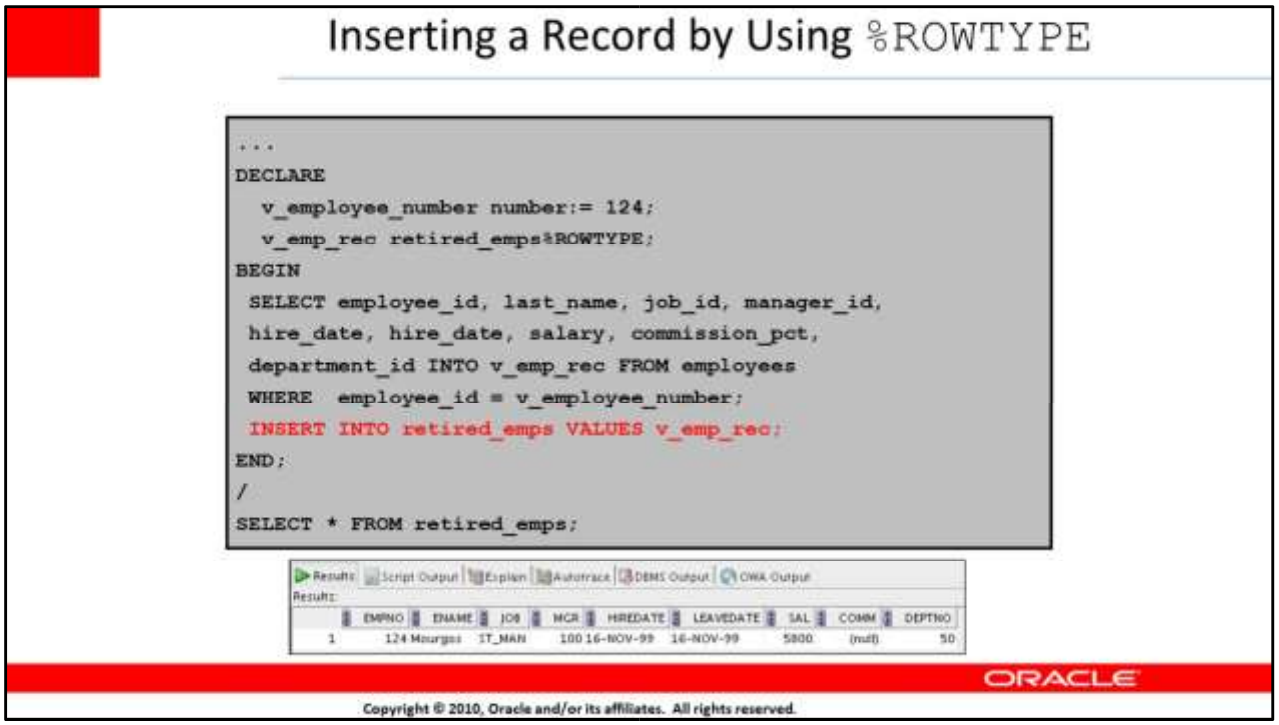
Note

The record that is inserted into the retired_emps table is shown in

the slide.

To see the output shown in the slide, place your cursor on the SELECT statement at the bottom of the code example in SQL Developer and press F9.

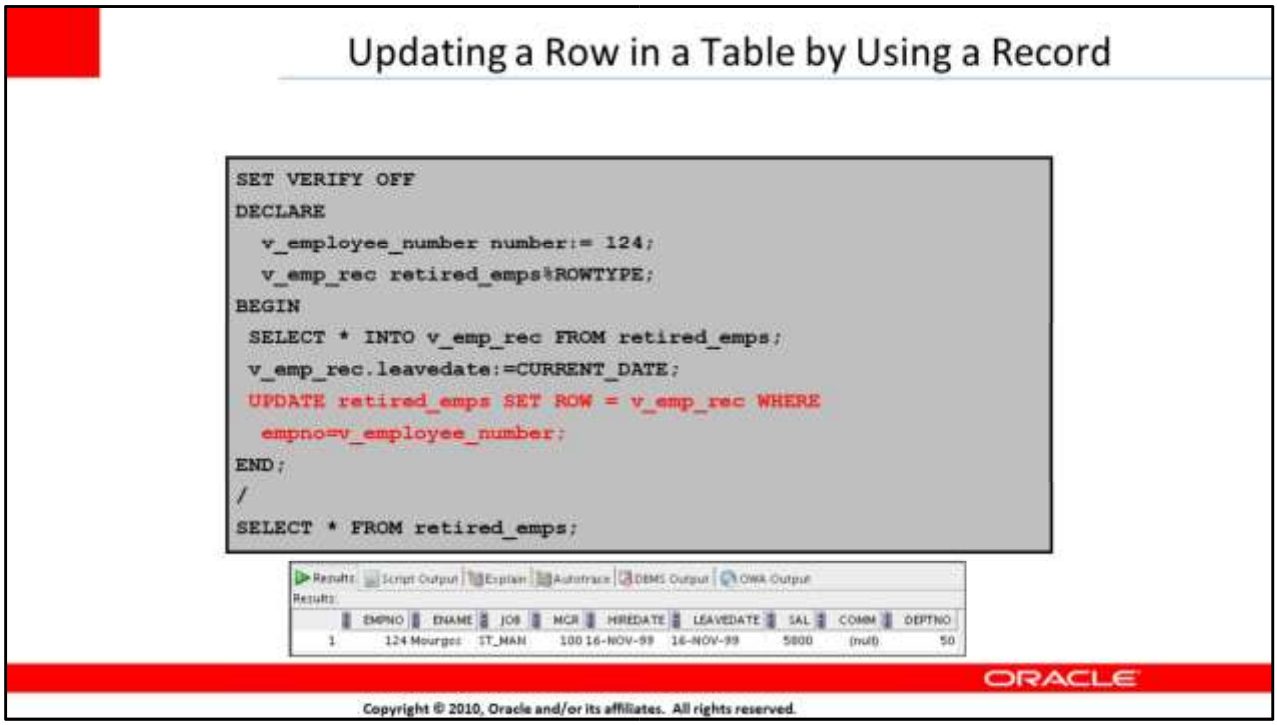
The complete code example is found in code_6_14_n-s.sql.



Inserting a Record by Using %ROWTYPE

Compare the INSERT statement in the previous slide with the INSERT statement in this slide. The emp_rec record is of type retired_emps. The number of fields in the record must be equal to the number of field names in the INTO clause. You can use this record to insert values into a table. This makes the code more readable.

Examine the SELECT statement in the slide. You select hire_date twice and insert the hire_date value in the leavedate field of retired_emps. No employee retires on the hire date. The inserted record is shown in the slide. (You will see how to update this in the next slide.) Note: To see the output shown in the slide, place your cursor on the SELECT statement at the bottom of the code example in SQL Developer and press F9.



Updating a Row in a Table by Using a Record

You learned to insert a row by using a record. This slide shows you how to update a row by using a record.

The ROW keyword is used to represent the entire row. The code shown in the slide updates the leavedate of the employee.

The record is updated as shown in the slide.

Note: To see the output shown in the slide, place your cursor on the SELECT statement at the bottom of the code example in SQL Developer and press F9.