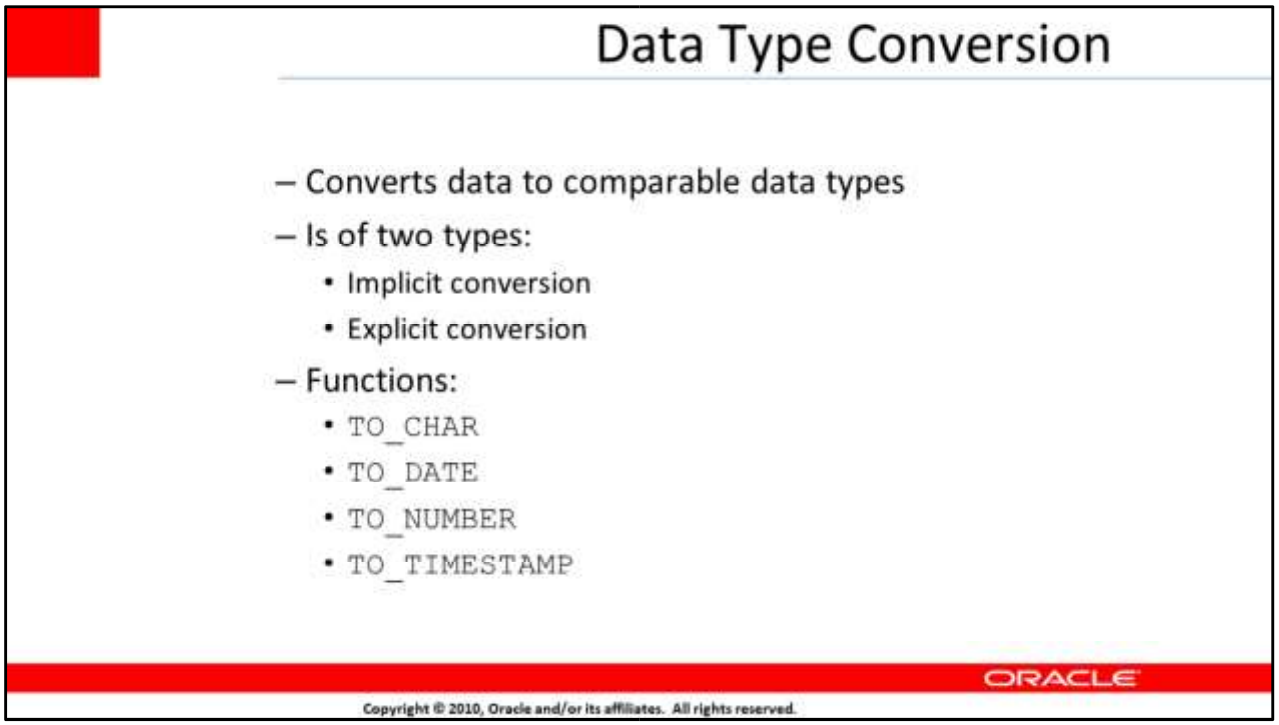


Data Type Conversion

In any programming language, converting one data type to another is a common requirement. PL/SQL can handle such conversions with scalar data types. Data type conversions can be of two types:

Implicit conversions: PL/SQL attempts to convert data types dynamically if they are mixed in a statement. Consider the following example:

```
DECLARE v_salary
 NUMBER(6):=6000; v_sal_hike
 VARCHAR2(5):='1000';
 v_total_salary v_salary%TYPE;
BEGIN
 v_total_salary:=v_salary + v_sal_hike;
END;
/
```

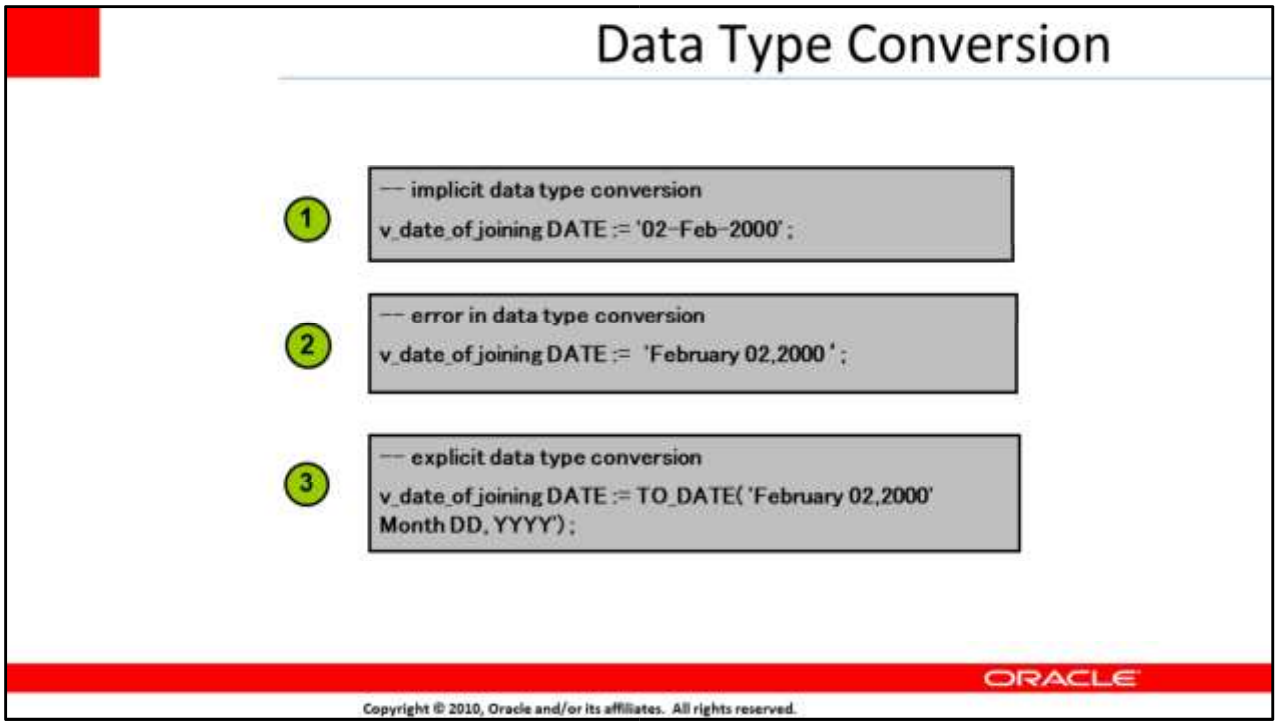
In this example, the sal_hike variable is of the VARCHAR2 type. When calculating the total salary, PL/SQL first converts sal_hike to NUMBER, and then performs the operation. The result is of the NUMBER type.

Implicit conversions can be between:

Characters and numbers

Characters and dates

Explicit conversions: To convert values from one data type to another, use built-in functions. For example, to convert a CHAR value to a DATE or NUMBER value, use TO_DATE or TO_NUMBER, respectively.



Data Type Conversion (continued)

Note the three examples of implicit and explicit conversions of the DATE data type in the slide:

1. Because the string literal being assigned to date_of_joining is in the default format, this example performs implicit conversion and assigns the specified date to date_of_joining.
2. The PL/SQL returns an error because the date that is being assigned is not in the default format.
3. The TO_DATE function is used to explicitly convert the given date in a particular format and assign it to the DATE data type variable date_of_joining.