



# Declaring PL/SQL Variables

# Objectives

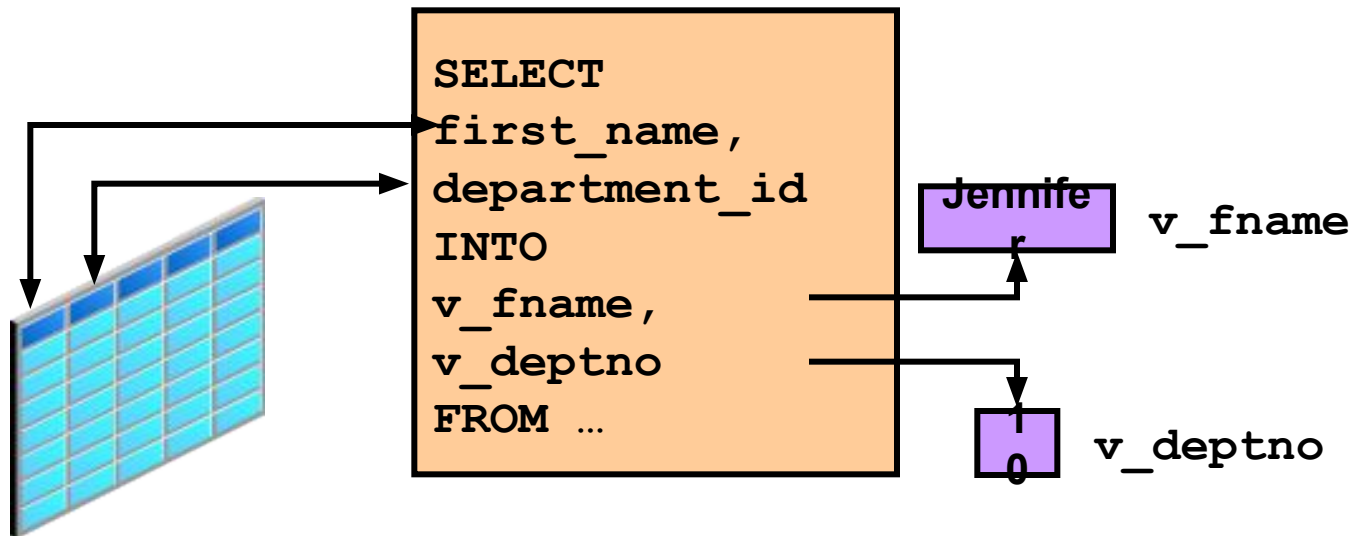
After completing this lesson, you should be able to do the following:

- Recognize valid and invalid identifiers
- List the uses of variables
- Declare and initialize variables
- List and describe various data types
- Identify the benefits of using the `%TYPE` attribute
- Declare, use, and print bind variables

# Use of Variables

Variables can be used for:

- Temporary storage of data
- Manipulation of stored values
- Reusability



# Requirements for Variable Names

A variable name:

- Must start with a letter
- Can include letters or numbers
- Can include special characters (such as \$, \_\_, and # )
- Must contain no more than 30 characters
- Must not include reserved words



# Handling Variables in PL/SQL

Variables are:

- Declared and initialized in the declarative section
- Used and assigned new values in the executable section
- Passed as parameters to PL/SQL subprograms
- Used to hold the output of a PL/SQL subprogram

# Declaring and Initializing PL/SQL Variables

Syntax:

```
identifier [CONSTANT] datatype [NOT NULL]  
    [:= | DEFAULT expr];
```

Examples:

```
DECLARE  
    v_hiredate    DATE;  
    v_deptno     NUMBER(2) NOT NULL := 10;  
    v_location   VARCHAR2(13) := 'Atlanta';  
    c_comm       CONSTANT NUMBER := 1400;
```

# Declaring and Initializing PL/SQL Variables

1

```
DECLARE
  v_myName VARCHAR2(20);
BEGIN
  DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
  v_myName := 'John';
  DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

2

```
DECLARE
  v_myName VARCHAR2(20) := 'John';
BEGIN
  v_myName := 'Steven';
  DBMS_OUTPUT.PUT_LINE('My name is: ' || v_myName);
END;
/
```

# Delimiters in String Literals

```
DECLARE
    v_event VARCHAR2(15);
BEGIN
    v_event := q'!Father's day!';
    DBMS_OUTPUT.PUT_LINE('3rd Sunday in June is :
    || v_event );
    v_event := q'[Mother's day]';
    DBMS_OUTPUT.PUT_LINE('2nd Sunday in May is :
    || v_event );
END;
/
```

```
anonymous block completed
3rd Sunday in June is : Father's day
2nd Sunday in May is : Mother's day
```



# Types of Variables

- PL/SQL variables:
  - Scalar
  - Composite
  - Reference
  - Large object (LOB)
- Non-PL/SQL variables: Bind variables

# Types of Variables

TRUE



25-JAN-01

Snow White

Long, long ago,  
in a land far, far away,  
there lived a princess called  
Snow White. . .

256120.08



Atlanta



# Guidelines for Declaring and Initializing PL/SQL Variables

- Follow naming conventions.
- Use meaningful identifiers for variables.
- Initialize variables designated as `NOT NULL` and `CONSTANT`.
- Initialize variables with the assignment operator (`:=`) or the `DEFAULT` keyword:

```
v_myName VARCHAR2 (20) := 'John' ;
```

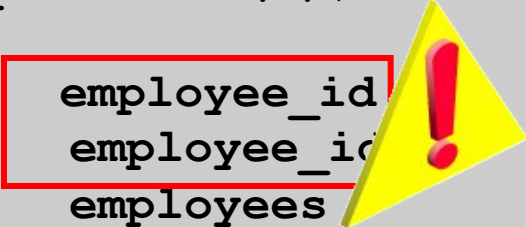
```
v_myName VARCHAR2 (20) DEFAULT 'John' ;
```

- Declare one identifier per line for better readability and code maintenance.

# Guidelines for Declaring PL/SQL Variables

- Avoid using column names as identifiers.

```
DECLARE
  employee_id NUMBER(6);
BEGIN
  SELECT      employee_id
             INTO      employee_id
             FROM      employees
             WHERE     last_name = 'Kochhar';
END;
/
```



- Use the NOT NULL constraint when the variable must hold a value.

# Scalar Data Types

- Hold a single value
- Have no internal components

TRUE

25-JAN-01

**The soul of the lazy man  
desires, and he has nothing;  
but the soul of the diligent  
shall be made rich.**

256120.08

Atlanta

# Base Scalar Data Types

- CHAR [(maximum\_length)]
- VARCHAR2 (maximum\_length)
- NUMBER [(precision, scale)]
- BINARY\_INTEGER
- PLS\_INTEGER
- BOOLEAN
- BINARY\_FLOAT
- BINARY\_DOUBLE



# Base Scalar Data Types

- DATE
- TIMESTAMP
- TIMESTAMP WITH TIME ZONE
- TIMESTAMP WITH LOCAL TIME ZONE
- INTERVAL YEAR TO MONTH
- INTERVAL DAY TO SECOND





# Declaring Scalar Variables

Examples:

```
DECLARE
  v_emp_job          VARCHAR2 (9) ;
  v_count_loop      BINARY_INTEGER := 0;
  v_dept_total_sal  NUMBER(9,2) := 0;
  v_orderdate       DATE := SYSDATE + 7;
  c_tax_rate        CONSTANT NUMBER(3,2) := 8.25;
  v_valid           BOOLEAN NOT NULL := TRUE;
  ...
```

## **%TYPE Attribute**

- Is used to declare a variable according to:
  - A database column definition
  - Another declared variable
- Is prefixed with:
  - The database table and column names
  - The name of the declared variable



# Declaring Variables with the %TYPE Attribute

## Syntax

```
identifier          table.column_name%TYPE;
```

## Examples

```
...  
  emp_lname          employees.last_name%TYPE;  
...
```

```
...  
  balance            NUMBER(7,2);  
  min_balance        balance%TYPE := 1000;  
...
```

# Declaring Boolean Variables

- Only the `TRUE`, `FALSE`, and `NULL` values can be assigned to a Boolean variable.
- Conditional expressions use the logical operators `AND` and `OR` and the unary operator `NOT` to check the variable values.
- The variables always yield `TRUE`, `FALSE`, or `NULL`.
- Arithmetic, character, and date expressions can be used to return a Boolean value.

# Bind Variables

Bind variables are:

- Created in the environment
- Also called *host variables*
- Created with the `VARIABLE` keyword
- Used in SQL statements and PL/SQL blocks
- Accessed even after the PL/SQL block is executed
- Referenced with a preceding colon





# Printing Bind Variables

Example:

```
VARIABLE b_emp_salary NUMBER
BEGIN
    SELECT salary INTO :b_emp_salary
    FROM employees WHERE employee_id = 178;
END;
/
PRINT b_emp_salary
SELECT first_name, last_name FROM employees
WHERE salary=:b_emp_salary;
```

# Printing Bind Variables

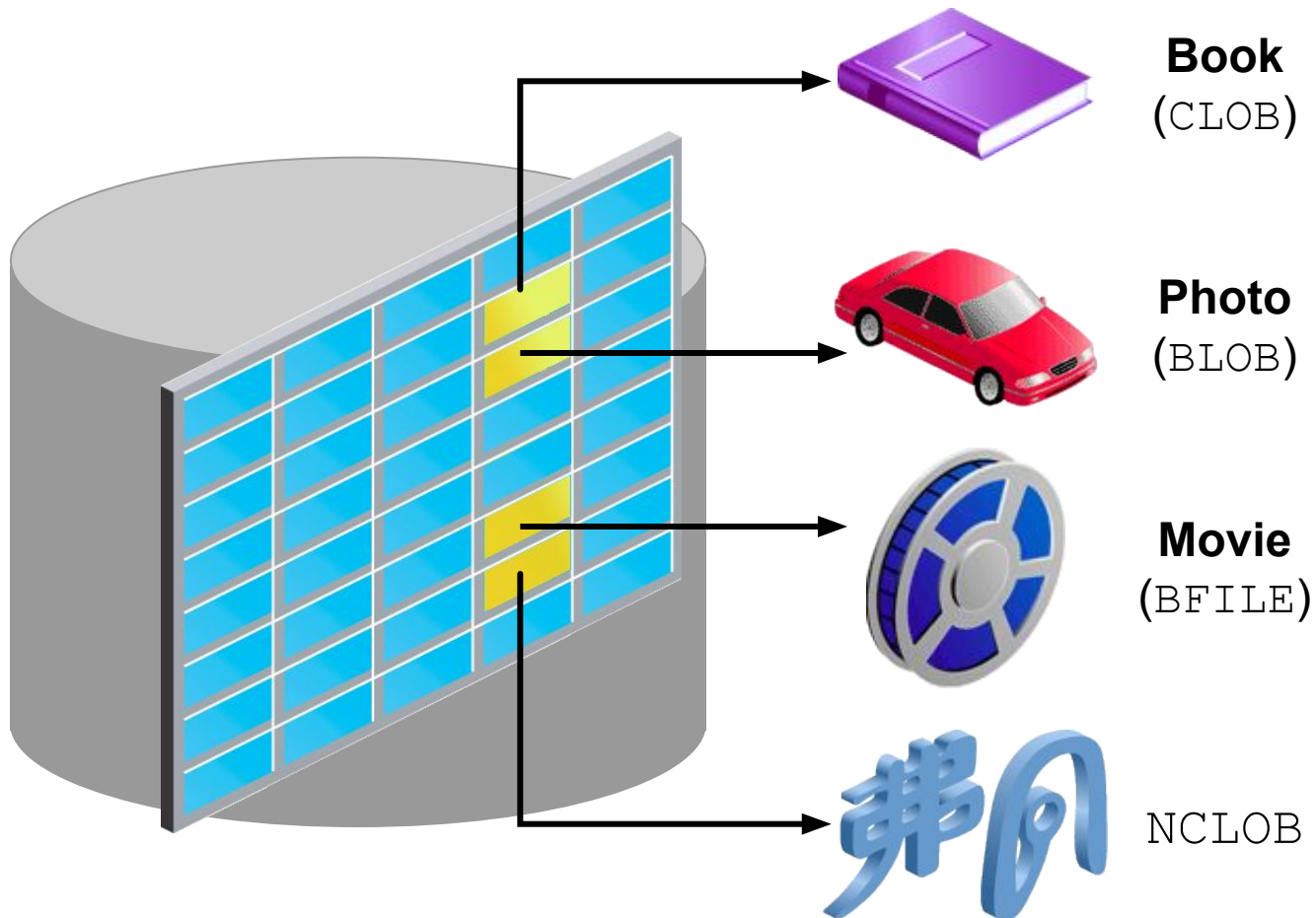
Example:

```
VARIABLE b_emp_salary NUMBER
SET AUTOPRINT ON
DECLARE
  v_empno NUMBER(6) := &empno;
BEGIN
  SELECT salary INTO :b_emp_salary
  FROM employees WHERE employee_id = v_empno;
END;
```


Output:

```
7000
```

# LOB Data Type Variables



# Composite Data Types

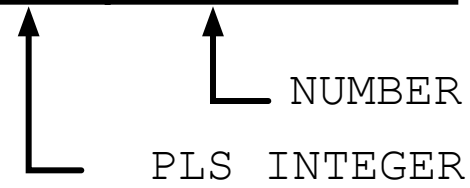
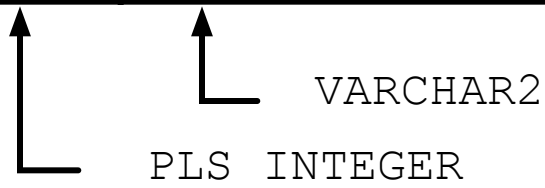
TRUE	23-DEC-98	ATLANTA	
------	-----------	---------	---

**PL/SQL table structure**

**PL/SQL table structure**

1	SMITH
2	JONES
3	NANCY
4	TIM

1	5000
2	2345
3	12
4	3456



# Quiz

The `%TYPE` attribute:

1. Is used to declare a variable according to a database column definition
2. Is used to declare a variable according to a collection of columns in a database table or view
3. Is used to declare a variable according the definition of another declared variable
4. Is prefixed with the database table and column names or the name of the declared variable

# Summary

In this lesson, you should have learned how to:

- Recognize valid and invalid identifiers
- Declare variables in the declarative section of a PL/SQL block
- Initialize variables and use them in the executable section
- Differentiate between scalar and composite data types
- Use the `%TYPE` attribute
- Use bind variables

# Practice 2: Overview

This practice covers the following topics:

- Determining valid identifiers
- Determining valid variable declarations
- Declaring variables within an anonymous block
- Using the `%TYPE` attribute to declare variables
- Declaring and printing a bind variable
- Executing a PL/SQL block







