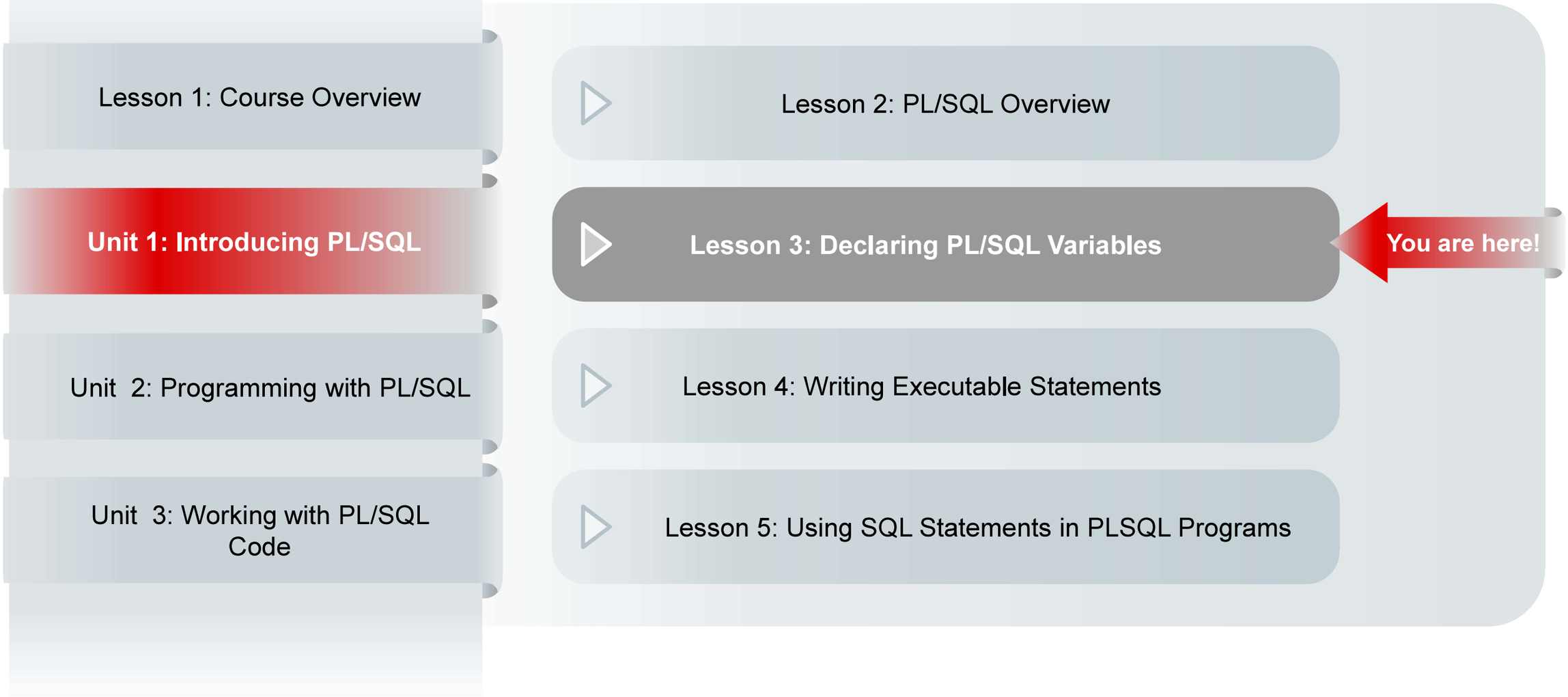
Declaring PLSQL Variables

Course Road Map



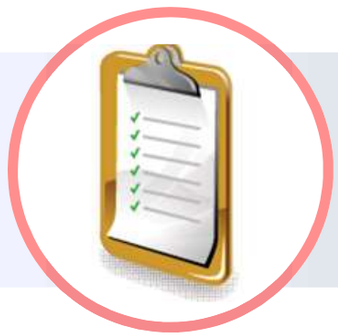
Objectives

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

Agenda

- Introducing variables
- Handling variables of different types
- Using the %TYPE attribute and composite data types
- Using bind variables



Variables

- Are labeled storage locations
- Are used to store and process data in a PL/SQL block
- Can hold different types of data
- Should declare variables before using them in the PL/SQL block

Variables in PL/SQL

Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort... Filter:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-11	AD_PRES	24000
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-09	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-09	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-14	IT_PROG	9000

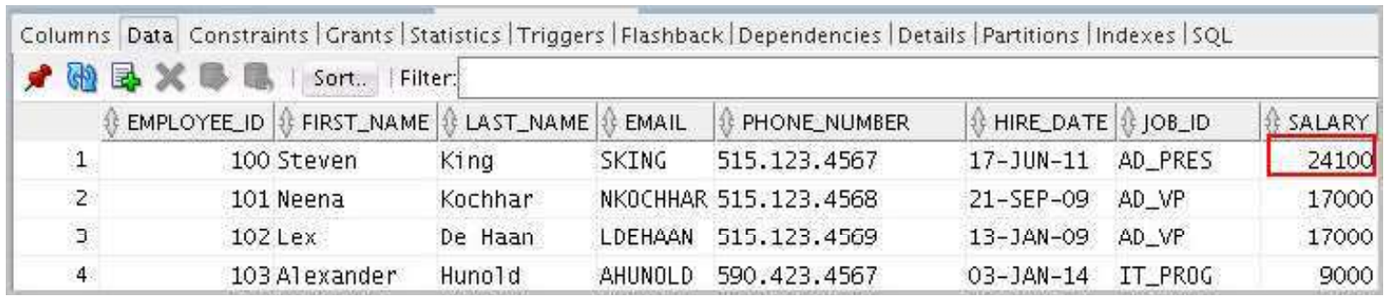
```
DECLARE
  v_sal NUMBER(8,2);
BEGIN
  SELECT salary INTO v_sal
  FROM employees
  WHERE employee_id = 100;
  UPDATE employees
  SET salary = v_sal+100
  WHERE employee_id = 100;
END;
```

Script Output ×

Task completed in 0.002 secon

```
PL/SQL procedure successfully completed.
```

Variables in PL/SQL



Columns | Data | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY
1	100	Steven	King	SKING	515.123.4567	17-JUN-11	AD_PRES	24100
2	101	Neena	Kochhar	NKOCHHAR	515.123.4568	21-SEP-09	AD_VP	17000
3	102	Lex	De Haan	LDEHAAN	515.123.4569	13-JAN-09	AD_VP	17000
4	103	Alexander	Hunold	AHUNOLD	590.423.4567	03-JAN-14	IT_PROG	9000

Requirements for Variable Names

•A variable name:

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

Using Variables in PL/SQL

- Variables are:
  - Declared and (optionally) initialized in the declaration 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:

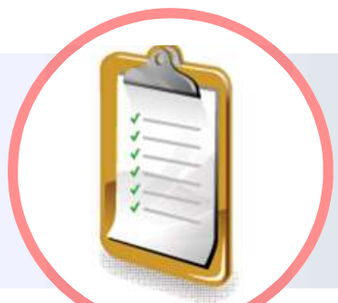
```
identifier [CONSTANT] datatype [NOT NULL]
     [:= | DEFAULT expr];
```

Examples:

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

Agenda

- Introducing variables
- Handling variables of different types
- Using the %TYPE attribute and composite data types
- Using bind variables



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;
/
```

Initializing Variables Through a SELECT Statement

- Retrieve data from the database with a SELECT statement.
- Syntax:

```
SELECT   select_list
INTO     {variable_name[, variable_name]...}
FROM     table
[WHERE   condition];
```

Types of Variables

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

Declaring 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;
   ...
```

Guidelines for Declaring and Initializing PL/SQL Variables

- Follow consistent naming conventions.
- Use meaningful identifiers for variables.
- Initialize variables that are 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.

Naming Conventions of the PL/SQL Structures Used

PL/SQL Structure	Convention	Example
Variable	v_variable_name	v_rate
Constant	c_constant_name	c_rate
Subprogram parameter	p_parameter_name	p_id
Bind (host) variable	b_bind_name	b_salary
Cursor	cur_cursor_name	cur_emp
Record	rec_record_name	rec_emp
Type	type_name_type	ename_table_type
Exception	e_exception_name	e_products_invalid
File handle	f_file_handle_name	f_file

Data Types for Strings

- A string is a sequence of symbols that are part of a character set. To declare a string variable, PL/SQL offers:
  - CHAR
  - NCHAR
  - VARCHAR
  - NVARCHAR
  - CLOB
  - NCLOB

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;
/
```

Resulting output

```
PL/SQL procedure successfully completed.

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

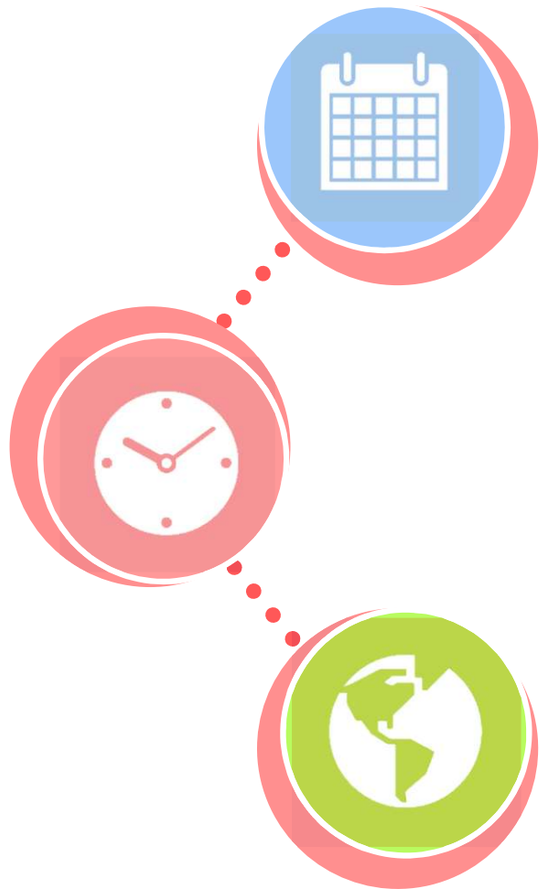
Data Types for Numeric values

- PL/SQL offers different numeric data types to suit different purposes:
  - NUMBER
  - PLS_INTEGER
  - SIMPLE_INTEGER
  - BINARY_INTEGER
  - BINARY_FLOAT
  - BINARY_DOUBLE
  - BOOLEAN (TRUE, FALSE or NULL)

BOOLEAN is a non-numeric data type, which can assume one of the three values – TRUE, FALSE or NULL

Data Types for Date and Time values

- The Oracle database provides three data types to work with dates and times:
  - DATE
  - TIMESTAMP WITH TIME ZONE
  - TIMESTAMP WITH LOCAL TIME ZONE
  - INTERVAL YEAR TO MONTH
  - INTERVAL DAY TO SECOND



Data Type Conversion

- Converts data to comparable data types
- Is of two types:
  - Implicit conversion
  - Explicit conversion
- Includes functions:
  - `TO_CHAR`
  - `TO_DATE`
  - `TO_NUMBER`
  - `TO_TIMESTAMP`

Data Type Conversion

1

```
-- implicit data type conversion
v_date_of_joining DATE:= '02-Feb-2000';
```

2

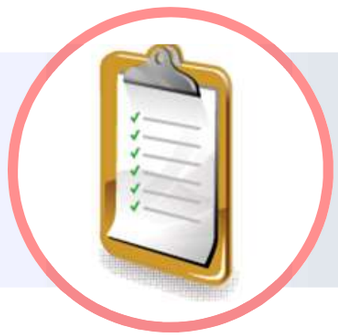
```
-- error in data type conversion
v_date_of_joining DATE:= 'February 02,2000';
```

3

```
-- explicit data type conversion
v_date_of_joining DATE:= TO_DATE('February 02,2000','Month
DD, YYYY');
```

Agenda

- Introducing variables
- Handling variables of different types
- Using the `%TYPE` attribute and composite data types
- Using bind variables



The %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 name
  - The name of the declared variable

Declaring Variables with the %TYPE Attribute

Syntax:

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

Examples:

```
...
  v_emp_lname       employees.last_name%TYPE;
...
```

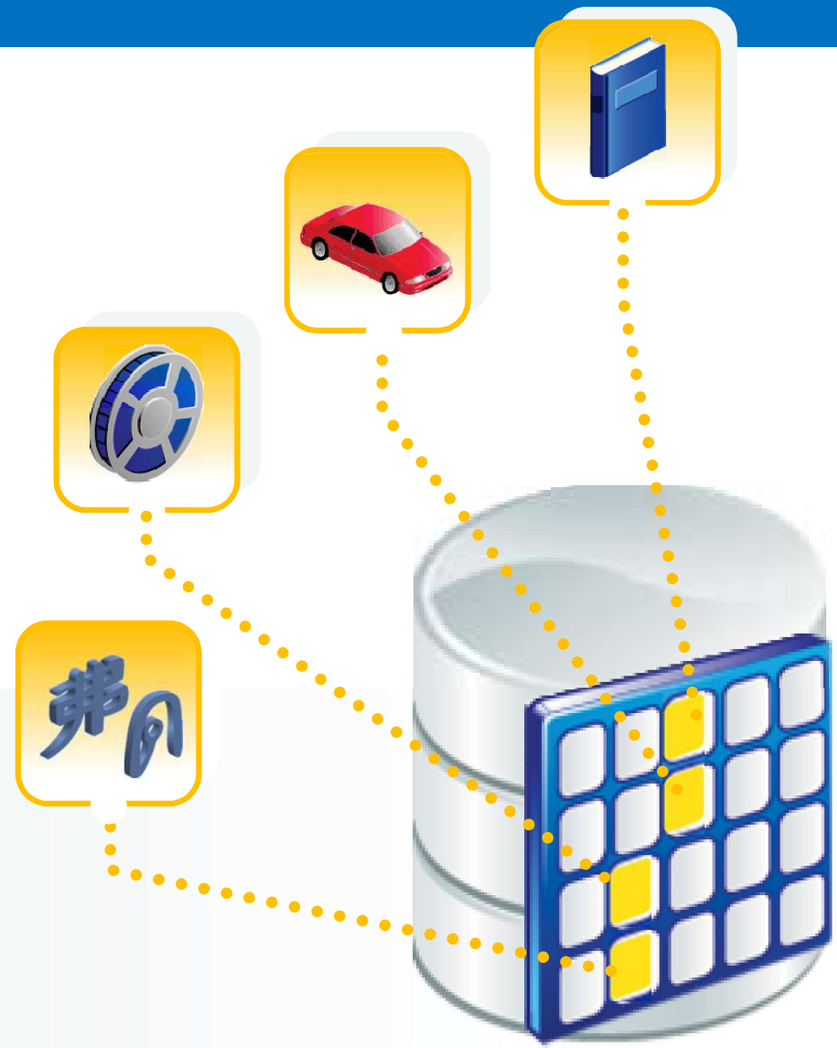
```
...
  v_balance         NUMBER(7,2);
  v_min_balance     v_balance%TYPE := 1000;
...
```

Declaring Boolean Variables

- Only `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.

LOB Data Type Variables

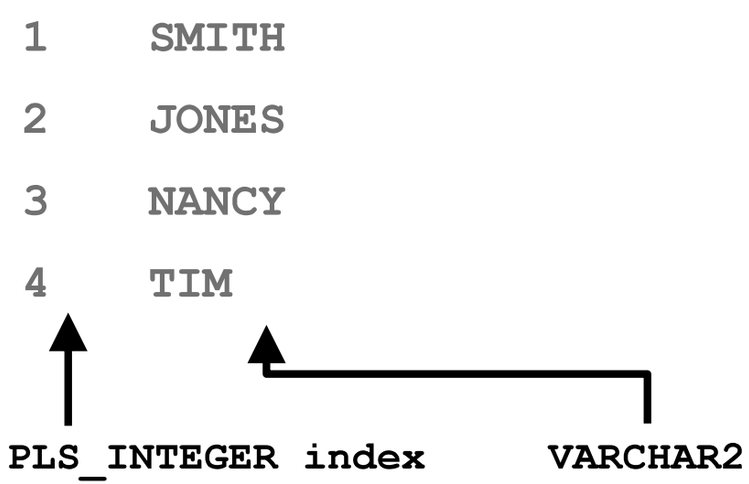
- Large objects (LOBs) are meant to store a large amount of data.
- LOB data types:
  - Character large object (CLOB)
  - Binary large object (BLOB)
  - Binary file (BFILE)
  - National language character large object (NCLOB)
- LOB data types enable you to store blocks of unstructured data up to 4 gigabytes in size.



Composite Data Types: Records and Collections

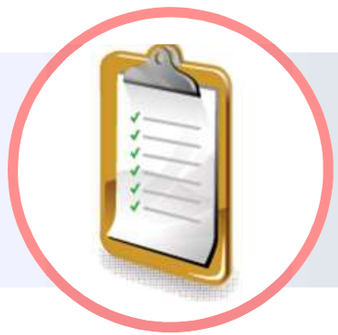
- PL/SQL Record:

PL/SQL Collections:



Agenda

- Introducing variables
- Handling variables of different types
- Using variable data types and the `%TYPE` attribute
- Using bind variables



Bind Variables

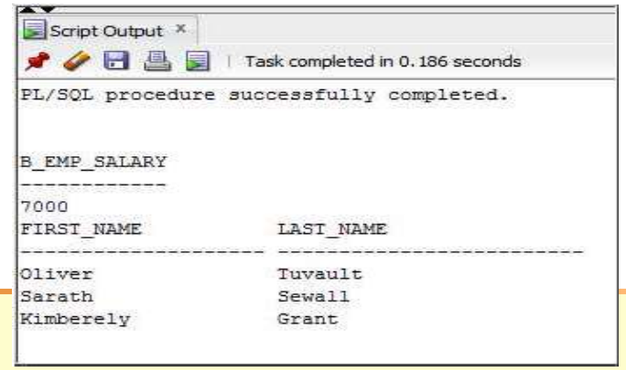
- Bind variables are:
  - Created in the host environment
  - Also called *host* variables
  - Created with the `VARIABLE` keyword in PL/SQL
  - Used in SQL statements and PL/SQL blocks
  - Accessed even after the PL/SQL block is executed
  - Referenced with a preceding colon
- Values can be output by using the `PRINT` command.
- Bind variables are required when using SQL*Plus and SQL Developer.

Bind Variables: Examples

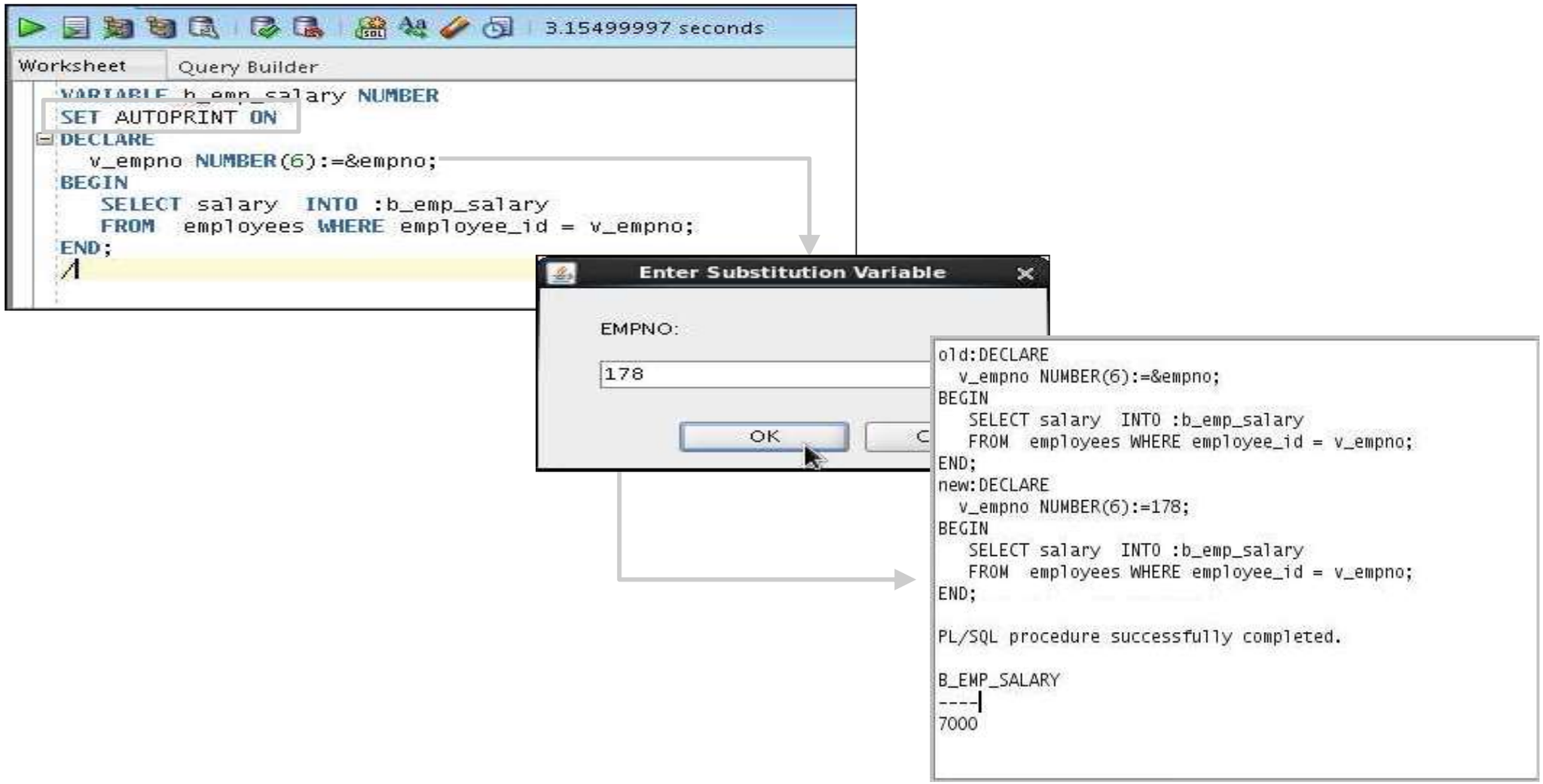
```
VARIABLE b_result NUMBER
BEGIN
  SELECT (SALARY*12) + NVL(COMMISSION_PCT,0) INTO
:b_result
  FROM employees WHERE employee_id = 144;
END;
/
PRINT b_result
```

```
 B_RESULT
----------
     30000
```

```
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;
```



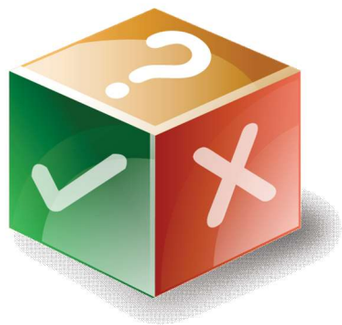
Using AUTOPRINT with Bind Variables



Quiz

The `%TYPE` attribute:

a. Is used to declare a variable according to a database column definition
b. Is used to declare a variable according to a collection of columns in a database table or view
c. Is used to declare a variable according to the definition of another declared variable
d. 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 3: 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