1 Declaring Variables

Objectives

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

- Recognize the basic PL/SQL block and its sections
- Describe the significance of variables in PL/SQL
- Declare PL/SQL variables
- Execute a PL/SQL block

ORACLE

PL/SQL Block Structure

DECLARE (Optional)

Variables, cursors, user-defined exceptions

BEGIN (Mandatory)

- SQL statements
- PL/SQL statements

EXCEPTION (Optional)

Actions to perform when errors occur

END; (Mandatory)

```
DECLARE
  ...
BEGIN
  ...
EXCEPTION
  ...
END;
```

ORACLE

Declaring Variables

```
SET SERVEROUTPUT ON
DECLARE
  v_chuoi VARCHAR2(20);
  v_ngay DATE;
BEGIN
  v_chuoi := 'Hom nay la ngay :';
  v_ngay := Sysdate;
  DBMS_OUTPUT.PUT_LINE (v_chuoi||v_ngay);
END;
```

ORACLE

Executing Statements and PL/SQL Blocks

```
DECLARE
  v_variable  VARCHAR2(5);
BEGIN
  SELECT column_name
  INTO v_variable
  FROM table_name;
EXCEPTION
  WHEN exception_name THEN
  ...
END;
```

```
DECLARE
 • • •
BEGIN
 • • •
EXCEPTION
 • • •
END;
```

```
DECLARE
  v_chuoi VARCHAR2(40);
  v_ngay DATE;
BEGIN
  SELECT last_name||' '||first_name, hire_date
      INTO v_chuoi, v_ngay
  FROM EMPLOYEES
  WHERE employee_id = &ID;
  DBMS_OUTPUT.PUT_LINE ('Nhan vien ' ||v_chuoi||' bat dau
                  lam viec ngay : ' ||v_ngay);
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    DBMS_OUTPUT.PUT_LINE ('KHONG CO NHAN VIEN CO');
END;
```

```
DECLARE
  invalid_so  EXCEPTION;
  v_so        integer := &so;
BEGIN
  IF v_so NOT IN (1,2,3) Then
    Raise invalid_so;
  ELSE
    DBMS_OUTPUT.PUT_LINE ('Gia tri : ' ||v_so);
  END IF;
EXCEPTION
  WHEN Invalid_so THEN
   DBMS_OUTPUT.PUT_LINE ('KHONG HOP LE');
END;
```

ORACLE

Block Types

Anonymous

```
[DECLARE]


BEGIN
   --statements

[EXCEPTION]

END;
```

Procedure

```
PROCEDURE name
IS

BEGIN
   --statements

[EXCEPTION]

END;
```

Function

```
FUNCTION name
RETURN datatype
IS
BEGIN
   --statements
   RETURN value;
[EXCEPTION]

END;
```

```
CREATE TABLE CHUSO(
SO NUMBER(1),
TENSO VARCHAR2(10));

INSERT INTO CHUSO VALUES (0, 'khong');
INSERT INTO CHUSO VALUES (1, 'mot');
INSERT INTO CHUSO VALUES (2, 'hai');
INSERT INTO CHUSO VALUES (3, 'ba');
INSERT INTO CHUSO VALUES (4, 'bon');
INSERT INTO CHUSO VALUES (5, 'nam');
INSERT INTO CHUSO VALUES (6, 'sau');
INSERT INTO CHUSO VALUES (7, 'bay');
INSERT INTO CHUSO VALUES (8, 'tam');
INSERT INTO CHUSO VALUES (9, 'chin');
```

```
CREATE FUNCTION so_chu (sodoi number)RETURN varchar2 IS
    dem number(2) := 1;
    num varchar2(50):= null;
    temp varchar2(10) := null;
    dai number(10) := 0;
BEGIN
    dai := length(sodoi);
    loop
        exit when dem > dai;
          select TENSO into temp from CHUSO
          where SO = substr(to_char(sodoi),dem,1);
          dem := dem + 1;
          num := num ||temp ||' ';
    end loop;
    return num;
END;
```

```
-- Thi hanh function
-- Cach 1
SELECT SO_CHU(9) FROM DUAL;
--Cach 2
VARIABLE TEMP VARCHAR2(100);
BEGIN
   :TEMP := SO_CHU(9);
END;
/
PRINT TEMP
-- Cach 3
SET SERVEROUTPUT ON
BEGIN
 DBMS_OUTPUT.PUT_LINE (SO_CHU(19));
END;
```

ORACLE

Program Constructs

```
DECLARE
  ...
BEGIN
  ...
EXCEPTION
  ...
END;
```

Tools Constructs
Anonymous blocks
Application procedures or functions
Application packages
Application triggers
Object types

Database Server Constructs
Anonymous blocks
Stored procedures or functions
Stored packages
Database triggers
Object types

Use of Variables

Variables can be used for:

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

ORACLE

Handling Variables in PL/SQL

- Declare and initialize variables in the declaration section.
- Assign new values to variables in the executable section.
- Pass values into PL/SQL blocks through parameters.
- View results through output variables.

ORACLE

Types of Variables

- PL/SQL variables:
  - Scalar
  - Composite
  - Reference
  - LOB (large objects)
- Non-PL/SQL variables: Bind and host variables

ORACLE

Using *i*SQL*Plus Variables Within PL/SQL Blocks

- PL/SQL does not have input or output capability of its own.
- You can reference substitution variables within a PL/SQL block with a preceding ampersand.
- *i*SQL*Plus host (or “bind”) variables can be used to pass run time values out of the PL/SQL block back to the *i*SQL*Plus environment.

ORACLE

Types of Variables

TRUE

25-JAN-01

"Four score and seven years ago our fathers brought forth upon this continent, a new nation, conceived in LIBERTY, and dedicated to the proposition that all men are created equal."

256120.08

Atlanta

Declaring 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;
```

ORACLE

Guidelines for Declaring PL/SQL Variables

- Follow naming conventions.
- Initialize variables designated as NOT NULL and CONSTANT.
- Declare one identifier per line.
- Initialize identifiers by using the assignment operator (:=) or the DEFAULT reserved word.

```
identifier := expr;
```

ORACLE

Naming Rules

- Two variables can have the same name, provided they are in different blocks.
- The variable name (identifier) should not be the same as the name of table columns used in the block.

```
t serveroutput on
ECLARE
 employee_id NUMBER(6);
EGIN
 SELECT   employee_id
 INTO     employee_id
 FROM employees
 WHERE last_name = 'Kochhar';
bms_output.put_line('aaa'||employee_id);
ND;
```

Adopt a naming convention for PL/SQL identifiers: for example, v_employee_id

ORACLE

Variable Initialization and Keywords

- Assignment operator (:=)
- DEFAULT keyword
- NOT NULL constraint

Syntax:

identifier := *expr*;

Examples:

```
v_hiredate := '01-JAN-2001';
```

```
v_ename := 'Maduro';
```

ORACLE

Scalar Data Types

- Hold a single value
- Have no internal components

25-OCT-99

256120.08

"Four score and seven years ago our fathers brought forth upon this continent, a new nation, conceived in LIBERTY, and dedicated to the proposition that all men are created equal"

TRUE

Atlanta

Base Scalar Data Types

- **CHAR [(*maximum_length*)]**
- **VARCHAR2 (*maximum_length*)**
- **LONG (**Base type for variable-length character data up to 32,760 bytes.)
- **LONG RAW (**Base type for binary data and byte strings up to 32,760 bytes.)
- **NUMBER [(*precision, scale*)]**
- **BINARY_INTEGER (**Base type for integers between -2,147,483,647 and 2,147,483,647.)
- **PLS_INTEGER (**Base type for signed integers between -2,147,483,647 and 2,147,483,647. PLS_INTEGER values require less storage and are faster than NUMBER and BINARY_INTEGER values.
- **BOOLEAN**

ORACLE

Base Scalar Data Types

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

ORACLE

Scalar Variable Declarations

Examples:

```
DECLARE
   v_job VARCHAR2(9);
   v_count BINARY_INTEGER := 0;
   v_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;
   ...
```

ORACLE

The %TYPE Attribute

- Declare a variable according to:
  - A database column definition
  - Another previously declared variable
- Prefix %TYPE with:
  - The database table and column
  - The previously declared variable name

ORACLE

Declaring Variables with the %TYPE Attribute

Syntax:

```
identifier        Table.column_name%TYPE;
```

Examples:

```
...
  v_name            employees.last_name%TYPE;
  v_balance         NUMBER(7,2);
  v_min_balance     v_balance%TYPE := 10;
...
```

Declaring Boolean Variables

- Only the values TRUE, FALSE, and NULL can be assigned to a Boolean variable.
- The variables are compared by the logical operators AND, OR, and NOT.
- The variables always yield TRUE, FALSE, or NULL.
- Arithmetic, character, and date expressions can be used to return a Boolean value.

ORACLE

Composite Data Types

TRUE	23-DEC-98	ATLANTA	

PL/SQL table structure

1	SMITH
2	JONES
3	NANCY
4	TIM

BINARY_INTEGER

VARCHAR2

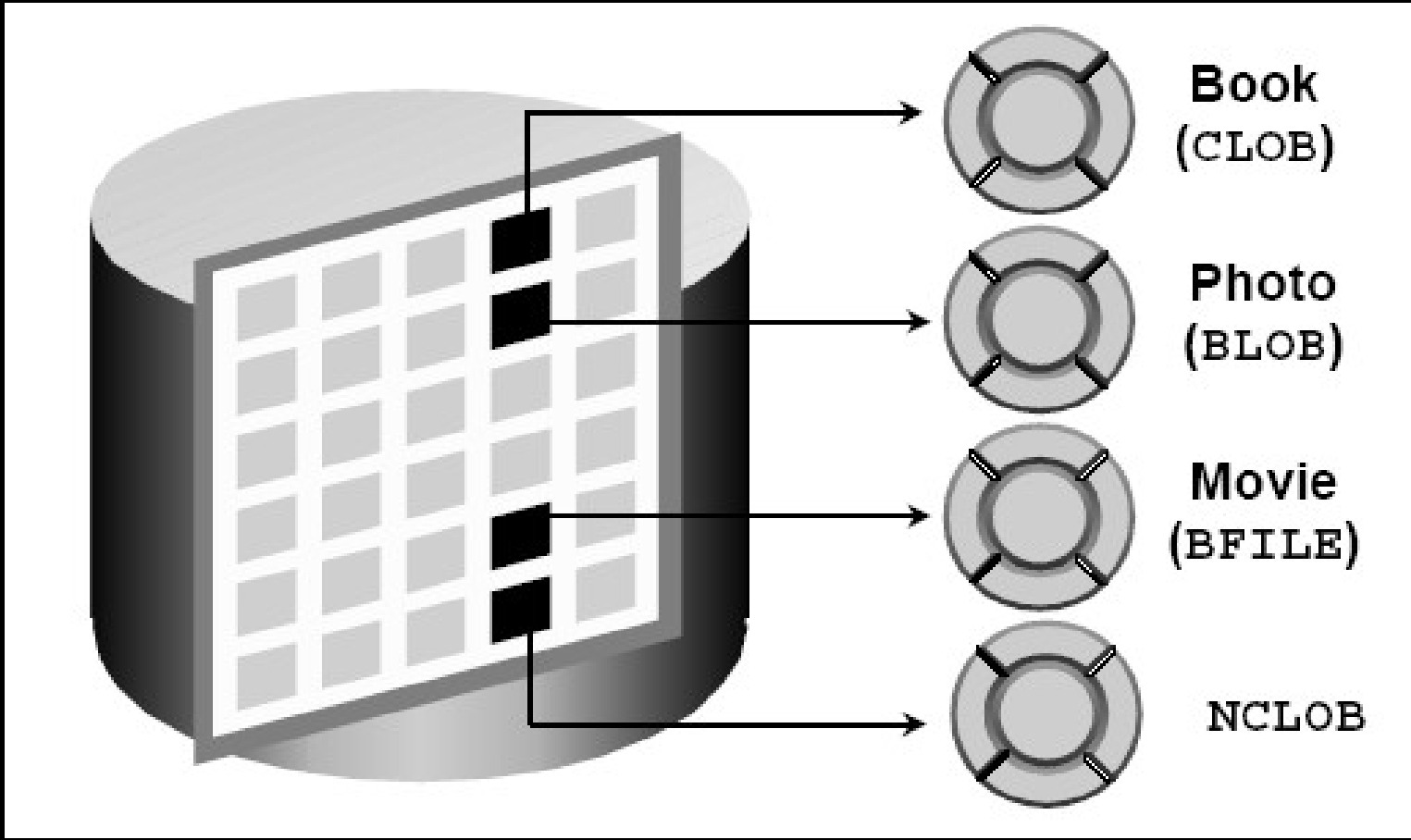
PL/SQL table structure

1	5000
2	2345
3	12
4	3456

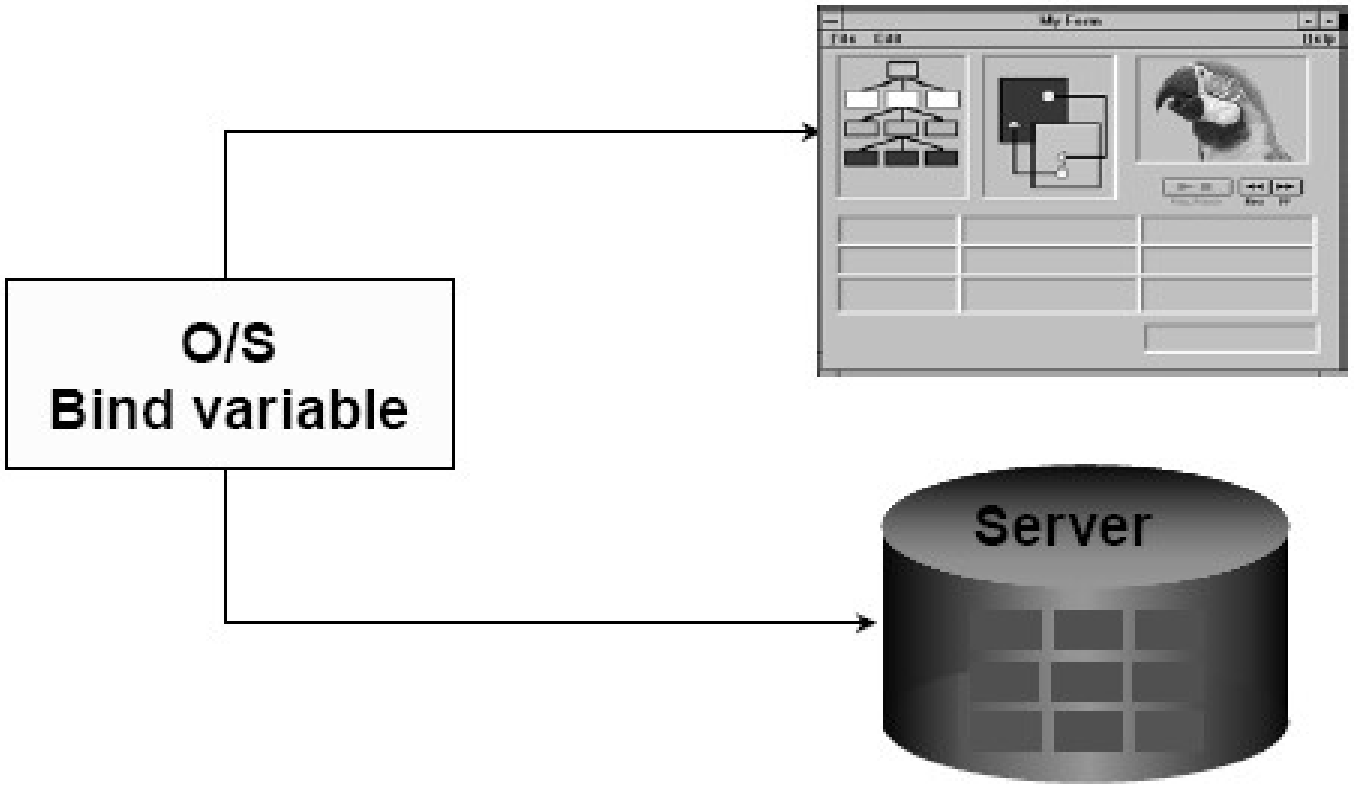
BINARY_INTEGER

NUMBER

LOB Data Type Variables



Bind Variables



Using Bind Variables

To reference a bind variable in PL/SQL, you must prefix its name with a colon (:).

Examples:

```
VARIABLE g_salary NUMBER
BEGIN
        SELECT      salary
        INTO                :g_salary
        FROM                employees
        WHERE       employee_id = 178;
END;
/
PRINT g_salary
```

ORACLE

Referencing Non-PL/SQL Variables

Store the annual salary into a *i*SQL*Plus host variable.

```
:g_monthly_sal := v_sal / 12;
```

- Reference non-PL/SQL variables as host variables.
- Prefix the references with a colon (:).

ORACLE

```
VARIABLE g_monthly_sal NUMBER
SET VERIFY OFF

DECLARE
v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
:g_monthly_sal := v_sal/12;
END;
/
PRINT g_monthly_sal
```

DBMS_OUTPUT.PUT_LINE

- An Oracle-supplied packaged procedure
- An alternative for displaying data from a PL/SQL block
- Must be enabled in *i*SQL*Plus with

  SET SERVEROUTPUT ON

```
SET SERVEROUTPUT ON
DEFINE p_annual_sal = 60000
```

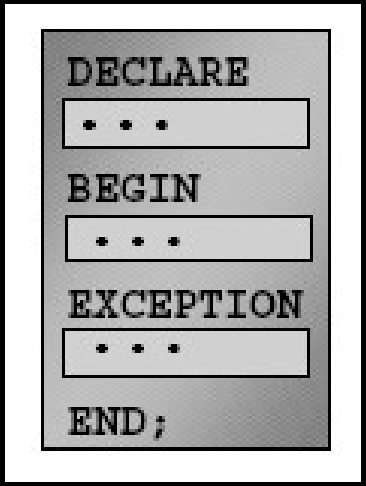
```
DECLARE
  v_sal NUMBER(9,2) := &p_annual_sal;
BEGIN
  v_sal := v_sal/12;
  DBMS_OUTPUT.PUT_LINE ('The monthly salary is ' ||TO_CHAR(v_sal));
END;
/
```

ORACLE

Summary

In this lesson you should have learned that:

- PL/SQL blocks are composed of the following sections:
  - Declarative (optional)
  - Executable (required)
  - Exception handling (optional)
- A PL/SQL block can be an anonymous block, procedure, or function.



Summary

In this lesson you should have learned that:

- PL/SQL identifiers:
  - Are defined in the declarative section
  - Can be of scalar, composite, reference, or LOB data type
  - Can be based on the structure of another variable or database object
  - Can be initialized
- Variables declared in an external environment such as *i*SQL*Plus are called host variables.
- Use DBMS_OUTPUT.PUT_LINE to display data from a PL/SQL block.

ORACLE

Practice 1 Overview

This practice covers the following topics:

- Determining validity of declarations
- Declaring a simple PL/SQL block
- Executing a simple PL/SQL block

ORACLE