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

PL/SQL Block Structure

DECLARE (Optional)

Variables, cursors, user-defined exceptions

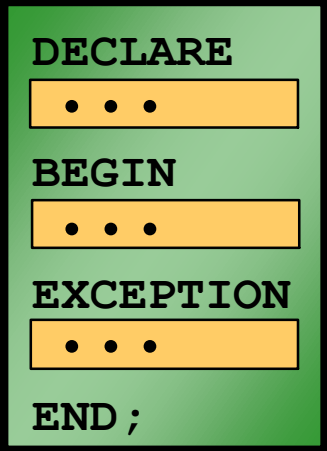
BEGIN (Mandatory)

- SQL statements
- PL/SQL statements

EXCEPTION (Optional)

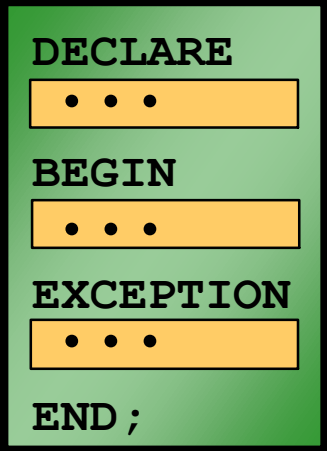
Actions to perform when errors occur

END; (Mandatory)



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;
```



Block Types

Anonymous

```
[DECLARE]

BEGIN
  --statements

[EXCEPTION]

END;
```

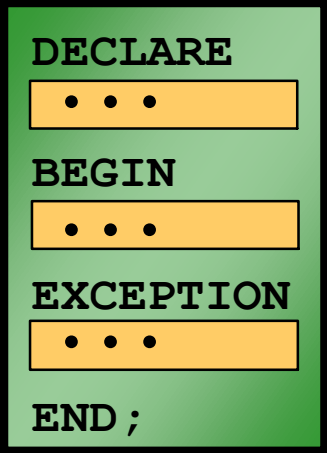
Procedure

```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

Function

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

END;
```

Program Constructs



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

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.

Types of Variables

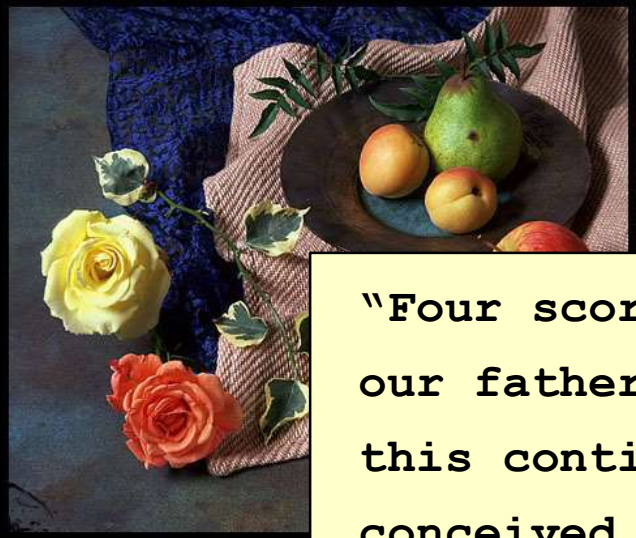
- PL/SQL variables:
  - Scalar
  - Composite
  - Reference
  - LOB (large objects)
- Non-PL/SQL variables: Bind and host variables

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.

Types of Variables

TRUE



25-JAN-01

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."

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;
```

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;
```

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.

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

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

Variable Initialization and Keywords

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

Syntax:

```
identifier := expr;
```

Examples:

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

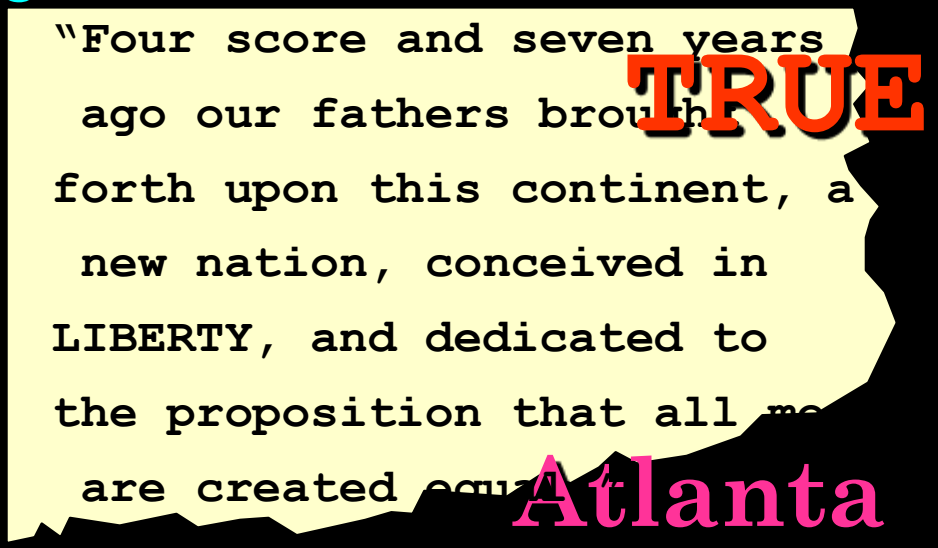
```
v_ename := 'Maduro';
```

Scalar Data Types

- Hold a single value
- Have no internal components

25-OCT-99

256120.08



Base Scalar Data Types

- CHAR [(*maximum_length*)]
- VARCHAR2 (*maximum_length*)
- LONG
- LONG RAW
- NUMBER [(*precision, scale*)]
- BINARY_INTEGER
- PLS_INTEGER
- BOOLEAN

Base Scalar Data Types

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

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;
  ...
```

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

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.

Composite Data Types

TRUE	23-DEC-98	ATLANTA	

PL/SQL table structure

1	SMITH
2	JONES
3	NANCY
4	TIM

BINARY_INTEGER

VARCHAR2

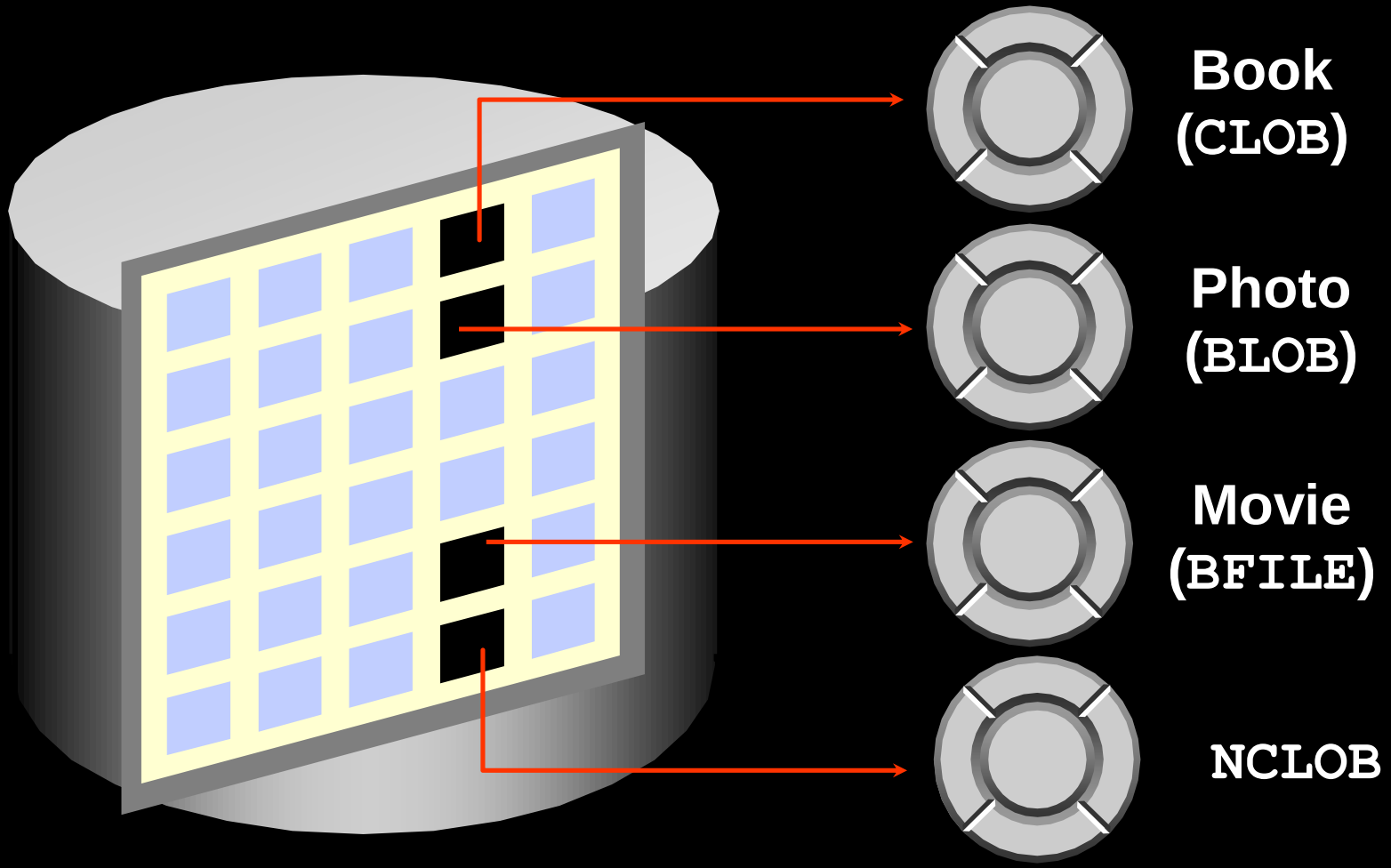
PL/SQL table structure

1	5000
2	2345
3	12
4	3456

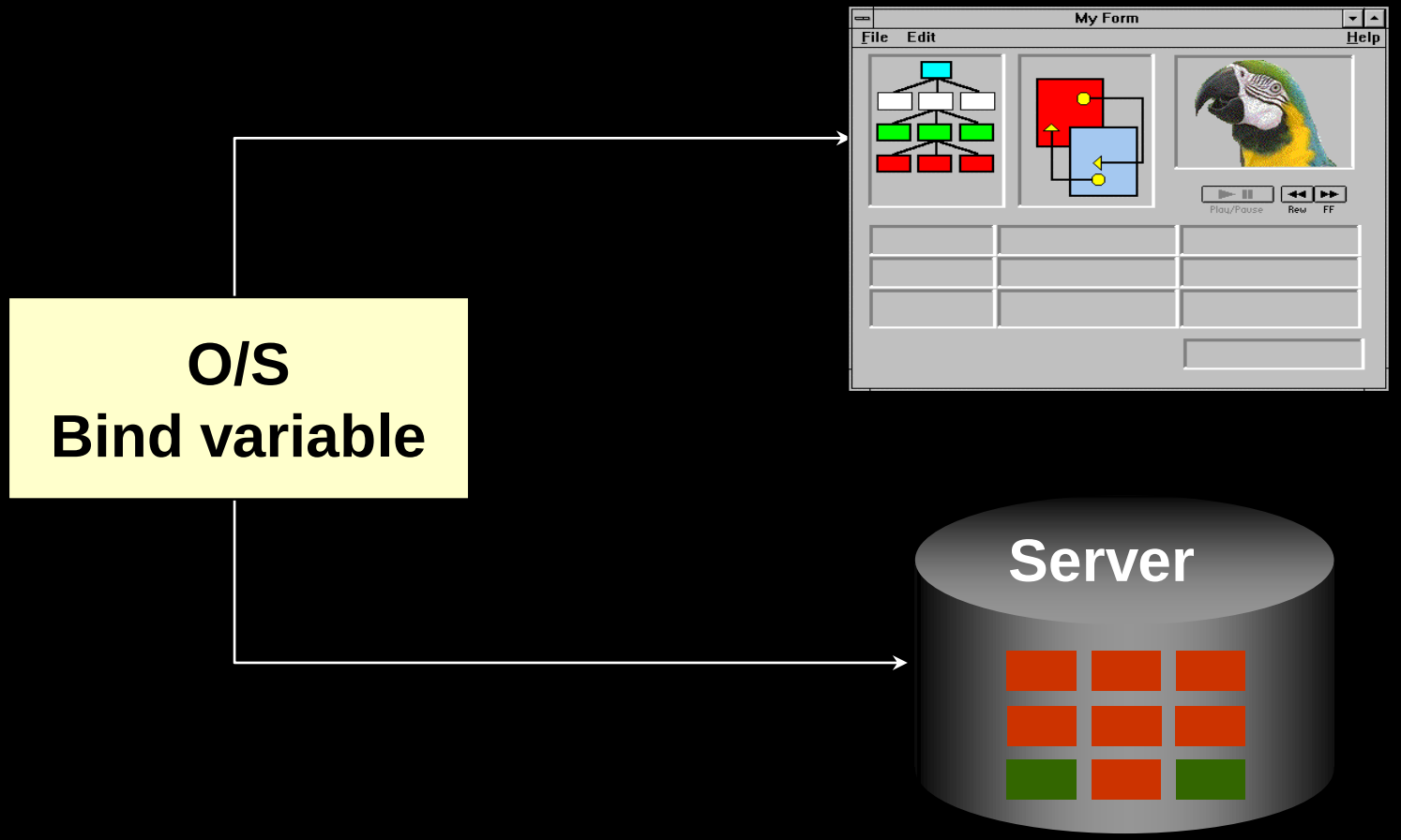
BINARY_INTEGER

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 (:).

Example:

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

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 (:).**

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`

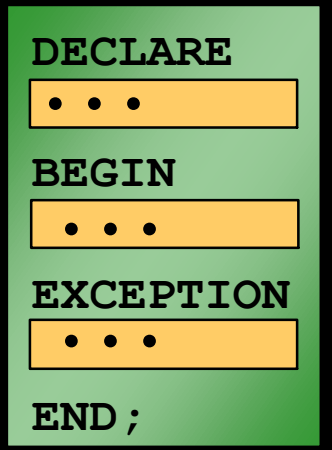
```
SET SERVEROUTPUT ON
DEFINE p_annual_sal = 60000
```

```
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;
/
```

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.

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