Declaring Variables

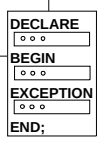
Objectives

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**

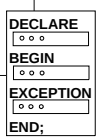
PL/SQL Block Structure

- **DECLARE – Optional**
  Variables, cursors, user-defined exceptions
- **BEGIN – Mandatory**
  - **SQL statements**
  - **PL/SQL statements**
- **EXCEPTION – Optional**
  Actions to perform when errors occur
- **END; – Mandatory**



PL/SQL Block Structure

```
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;
```

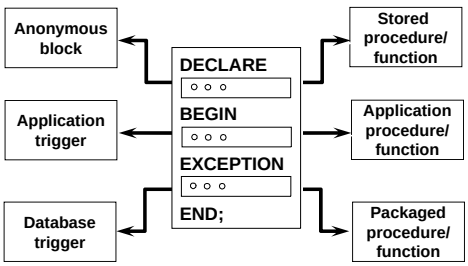
Procedure

```
PROCEDURE name
IS

BEGIN
  --statements

[EXCEPTION]

END;
```

Function

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

END;
```

Program Constructs



Use of Variables

Use variables 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.

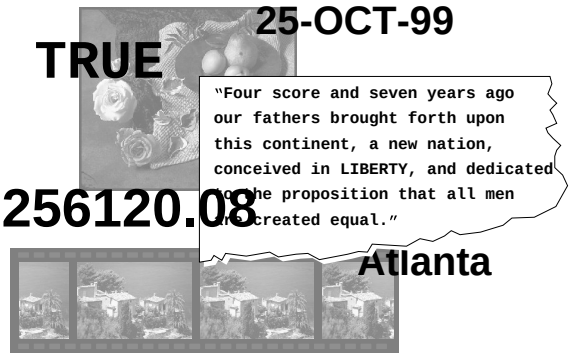
Types of Variables

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

Types of Variables

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

Types of Variables



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;
```

Declaring PL/SQL Variables

Guidelines

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

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
  empno  NUMBER(4);
BEGIN
  SELECT   empno
  INTO     empno
  FROM     emp
  WHERE    ename = 'SMITH';
END;
```

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

Assigning Values to Variables

Syntax

```
identifier := expr;
```

Examples

Set a predefined hiredate for new employees.

```
v_hiredate := '31-DEC-98';
```

Set the employee name to Maduro.

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

Variable Initialization and Keywords

Using:

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

Scalar Datatypes

- Hold a single value
- Have no internal components

25-OCT-99

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

TRUE

256120.08

Atlanta

Base Scalar Datatypes

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

Base Scalar Datatypes

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

Scalar Variable Declarations

Examples

```
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

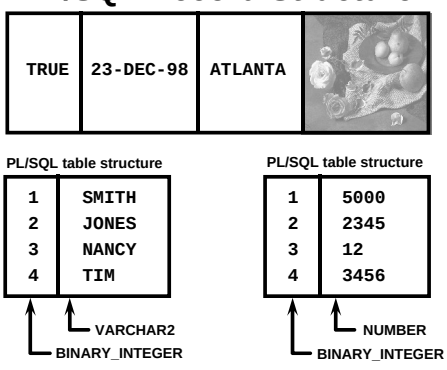
Examples

```
...
  v_ename              emp.ename%TYPE;
  v_balance            NUMBER(7,2);
  v_min_balance        v_balance%TYPE := 10;
...
```

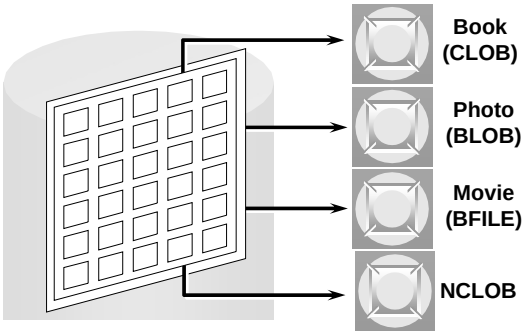
Declaring Boolean Variables

- Only the values TRUE, FALSE, and NULL can be assigned to a Boolean variable.
- The variables are connected 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.

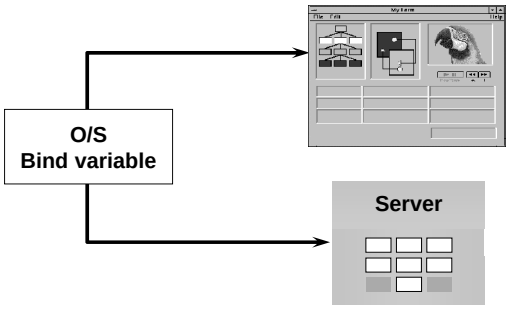
PL/SQL Record Structure



LOB Datatype Variables



Bind Variables



Referencing Non-PL/SQL Variables

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

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

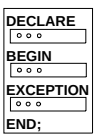
- Reference non-PL/SQL variables as host variables.
- Prefix the references with a colon (:).

DBMS_OUTPUT.PUT_LINE

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

Summary

- 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

- PL/SQL identifiers:
  - Are defined in the declarative section
  - Can be of scalar, composite, reference, or LOB datatype
  - Can be based on the structure of another variable or database object
  - Can be initialized