

PL/SQL Block Syntax and Guidelines

- **Statements can continue over several lines.**
- **Lexical units can be separated by:**
 - **Spaces**
 - **Delimiters**
 - **Identifiers**
 - **Literals**
 - **Comments**

PL/SQL Block Syntax and Guidelines

Identifiers

- Can contain up to 30 characters
- Cannot contain reserved words unless enclosed in double quotation marks
- Must begin with an alphabetic character
- Should not have the same name as a database table column name

PL/SQL Block Syntax and Guidelines

Literals

- Character and date literals must be enclosed in single quotation marks.

```
v_ename := 'Henderson';
```

- Numbers can be simple values or scientific notation.

Commenting Code

- Prefix single-line comments with two dashes (--).
- Place multi-line comments between the symbols /* and */.

Example

```
...
  v_sal NUMBER (9,2);
BEGIN
  /* Compute the annual salary based on the
     monthly salary input from the user */
  v_sal := &p_monthly_sal * 12;
END; -- This is the end of the transaction
```

SQL Functions in PL/SQL

- **Available:**

- **Single-row number**
- **Single-row character**
- **Datatype conversion**
- **Date**



Same as in SQL

- **Not available:**

- **DECODE**
- **Group functions**

PL/SQL Functions

Examples

- **Build the mailing list for a company.**

```
v_mailing_address := v_name || CHR(10) ||  
                    v_address || CHR(10) || v_state ||  
                    CHR(10) || v_zip;
```

- **Convert the employee name to lowercase.**

```
v_ename          := LOWER(v_ename);
```

Datatype Conversion

- Convert data to comparable datatypes.
- Mixed datatypes can result in an error and affect performance.
- Conversion functions:
 - TO_CHAR
 - TO_DATE
 - TO_NUMBER

```
DECLARE
    v_date VARCHAR2(15);
BEGIN
    SELECT TO_CHAR(hiredate,
                  'MON. DD, YYYY')
    INTO   v_date
    FROM   emp
    WHERE  empno = 7839;
END;
```

Datatype Conversion

This statement produces a compilation error if the variable `v_date` is declared as datatype `DATE`.

```
v_date := 'January 13, 1998';
```

To correct the error, use the `TO_DATE` conversion function.

```
v_date := TO_DATE ('January 13, 1998',  
                  'Month DD, YYYY');
```


Nested Blocks and Variable Scope

- **Statements can be nested wherever an executable statement is allowed.**
- **A nested block becomes a statement.**
- **An exception section can contain nested blocks.**
- **The scope of an object is the region of the program that can refer to the object.**

Nested Blocks and Variable Scope

An identifier is visible in the regions in which you can reference the unqualified identifier:

- **A block can look up to the enclosing block.**
- **A block cannot look down to enclosed blocks.**

Nested Blocks and Variable Scope

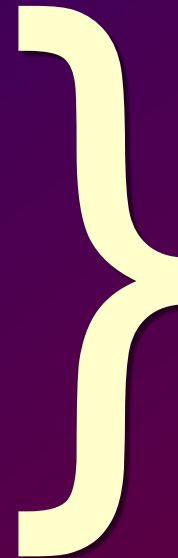
Example

```
...  
  x  BINARY_INTEGER;  
BEGIN  
  ...  
  DECLARE  
    y  NUMBER;  
  BEGIN  
    ...  
  END;  
  ...  
END;
```

The diagram shows two nested rectangles representing variable scopes. The outer rectangle, labeled "Scope of x", starts at the line "x BINARY_INTEGER;" and ends at the final "END;". The inner rectangle, labeled "Scope of y", starts at the line "y NUMBER;" and ends at the "END;" of the inner block. This visualizes that variable 'x' is visible throughout the entire code block, while variable 'y' is only visible within the inner block.

Operators in PL/SQL

- **Logical**
- **Arithmetic**
- **Concatenation**
- **Parentheses to control order of operations**
- **Exponential operator (**)**



**Same as in
SQL**

Operators in PL/SQL

Examples

- Increment the index for a loop.

```
v_count      := v_count + 1;
```

- Set the value of a Boolean flag.

```
v_equal      := (v_n1 = v_n2);
```

- Validate an employee number if it contains a value.

```
v_valid      := (v_empno IS NOT NULL);
```

Using Bind Variables

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

Example

```
VARIABLE g_salary NUMBER
DECLARE
    v_sal      emp.sal%TYPE;
BEGIN
    SELECT     sal
    INTO       v_sal
    FROM       emp
    WHERE      empno = 7369;
    :g_salary      := v_sal;
END;
/
```

Programming Guidelines

Make code maintenance easier by:

- **Documenting code with comments**
- **Developing a case convention for the code**
- **Developing naming conventions for identifiers and other objects**
- **Enhancing readability by indenting**

Code Naming Conventions

Avoid ambiguity:

- The names of local variables and formal parameters take precedence over the names of database tables.
- The names of columns take precedence over the names of local variables.

Indenting Code

For clarity, indent each level of code.

Example

```
BEGIN
  IF x=0 THEN
    y:=1;
  END IF;
END;
```

```
DECLARE
  v_deptno    NUMBER(2);
  v_location   VARCHAR2(13);
BEGIN
  SELECT  deptno,
         loc
  INTO    v_deptno,
         v_location
  FROM    dept
  WHERE   dname = 'SALES';

  ...
END;
```