


FACULTAD DE CIENCIAS Y TECNOLOGÍA

Ingeniería en Ciencias de la Computación
Ingeniería en Tecnologías de la Información y Seguridad



Capítulo VII

INTRODUCCION A LA PROGRAMACION PL/SQL

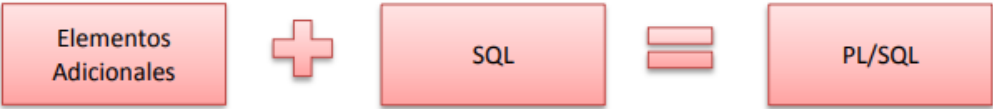
Ing. Edgar T. Espinoza R.

1

LENGUAJE DE PROGRAMACIÓN PL/SQL



- El lenguaje que se emplea para programar varía de un Sistema Manejador de Bases de Datos Relacional (RDBMS) a otro. No existe un estándar como en el caso del SQL.
- El lenguaje que utiliza Oracle se llama PL/SQL (Procedural Lenguaje extensión to SQL) y es un lenguaje de programación que se usa para acceder y trabajar con bases de datos en Oracle desde distintos entornos.



2

2



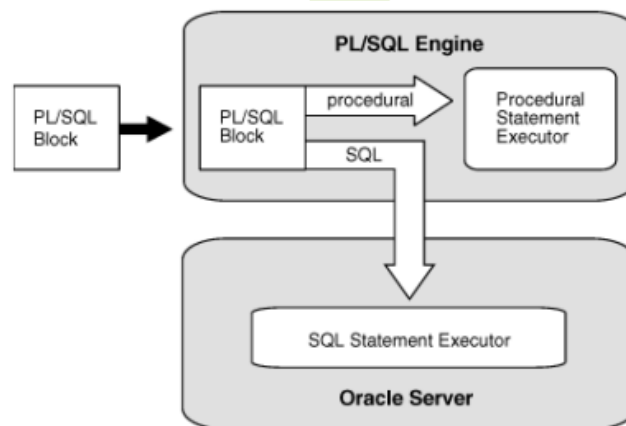
Principales elementos adicionales:

- Variables, constantes, tipos de datos.
- Estructuras de control: LOOPS, IF, ELSE, etc.
- Unidades o piezas de código PL/SQL reutilizables (escritas 1 vez, ejecutadas N veces)
- Características como encapsulamiento de datos, manejo de excepciones, ocultamiento de información, programación orientada a objetos, etc.

Cada programa PL/SQL puede contener uno o más bloques de código que pueden estar anidados

3

ARQUITECTURA PL/SQL



- Código PL/SQL es ejecutado en un “PL/SQL engine” a través de un “executor”.
- Este “engine” es implementado a través de una “Máquina virtual” encargado de procesar las instrucciones del código PL/SQL.
- Sentencias SQL son enviadas al servidor Oracle para ser ejecutadas por su propio “executor”.

4

Ventajas de la arquitectura



- Integración de estructuras procedurales con SQL
- Desempeño.
- En cuanto a programación:
 - Desarrollo modularizado.
 - Manejo adecuado de errores.
 - Portabilidad.
 - Integración con herramientas Oracle.

5

ELEMENTOS DE UN PROGRAMA PL/SQL



--sección declarativa (opcional)

[declare]

--ordenes sql y pl/sql (obligatorio)

begin

<instrucciones pl/sql>

--excepciones (opcional)

[exception]

--fin de bloque (obligatorio)

end;

/

Ejemplo

Programa PL/SQL mas simple

begin

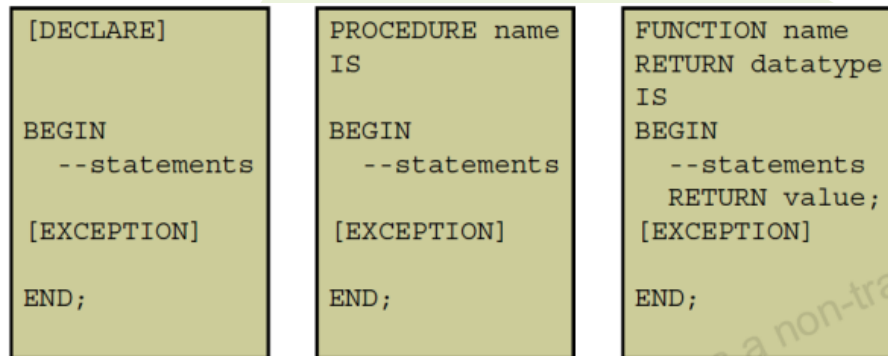
 null;

end;

/

6

Tipos de bloques PL/SQL



7

Bloques anónimos

- No contienen un nombre.
- No son almacenados en la BD.
- Al no ser almacenados, se compilan y se ejecutan cada vez que se requieran durante la ejecución de alguna aplicación.



Procedimientos

- Son objetos con nombre almacenados en la BD.
- No requieren ser re- compilados cada vez que se deseen ejecutar.

Funciones

- Similares a un procedimiento con la diferencia que una función regresa un valor de un determinado tipo de dato.

8

¿Qué se puede construir con bloques PL/SQL?



- Bloques anónimos
 - Embebidos en una aplicación
- Procedimientos almacenados.
 - Almacenados en la BD, se invocan de manera repetida por nombre, aceptan parámetros.
- Paquetes PL/SQL
 - Módulos que agrupan a un conjunto de procedimientos y/o funciones.
- Triggers
 - Asociados a una tabla y se ejecutan al ocurrir un evento.
- Objetos personalizados
 - Definidos por el usuario y almacenados en la BD

9

Ejemplo 'Hola Mundo' con PL/SQL



Crear un bloque PL/SQL anónimo que imprima el mensaje "Hola mundo!" y la fecha actual del sistema.

```
set serveroutput on
```

```
begin
```

```
    dbms_output.put_line('Hola mundo!, fecha actual: ' ||
                          sysdate);
```

```
end;
```

```
/
```

SERVEROUTPUT es una variable de ambiente de SQL *Plus que habilita la salida de mensajes a consola a través del uso de un buffer. De no activarse, los mensajes no se imprimirán en la consola.

10

VARIABLES, ASIGNACIONES Y OPERADORES.



Los tipos de datos que se emplean para declarar variables en un programa PL/SQL están integrados por los siguientes grupos:

- Todos los tipos de datos SQL.
- Tipos de datos particulares o específicos de PL/SQL

El alcance de una variable es local. Esto significa que una variable será visible únicamente en el bloque PL/SQL donde fue declarada.

Sintaxis:

`<identifier> [constant] <datatype> [not null] [:= | default] <expression>;`

11

Ejemplo:

```
set serveroutput on
```

```
declare
```

```
    v_uno number;
```

```
    v_dos number := 2;
```

```
    v_tres number not null := 3;
```

```
    v_cuatro number default 4;
```

```
    v_str varchar2(50);
```

```
Begin
```

```
    v_uno := 1;
```

```
    v_str := v_uno||','||v_dos||','||v_tres||','||v_cuatro;
```

```
    dbms_output.put_line(v_str);
```

```
end;
```

```
/
```



12

Otras formas de declarar variables



El atributo %type permite hacer referencia a una columna de una tabla o una variable que se haya definido anteriormente.

Sintaxis:

```
nombreVariable tabla.columna%type;
```

Ejemplo:

```
v_nombre empleado.nombre%type;
v_balance number;
```

Es posible hacer referencia a un renglón de una tabla o cursor, con el fin de crear variables que tengan que permitan acceder a todos los atributos o columnas de dicho renglón. para ello utilice emplea %rowtype.

```
nombrevariable {tabla | cursor}%rowtype;
```

13

Ejemplo:

Crear un Script PL/SQL que muestre en consola los datos del plan de estudios con id =1

```
declare
    v_id plan_estudios.plan_estudios_id%type;
    v_clave plan_estudios.clave%type;
    v_fecha plan_estudios.fecha_inicio%type;
begin
    select plan_estudios_id,clave,fecha_inicio
    into v_id,v_clave,v_fecha
    from plan_estudios
    where plan_estudios_id=1;
    dbms_output.put_line('id: ' || v_id);
    dbms_output.put_line('clave: ' || v_clave);
    dbms_output.put_line('fecha: ' || v_fecha);
end;
/
```



14

Convenciones para el nombrado de variables:



Estructura PL/SQL	Convención
Variable	v_variable_name
Constante	c_constant_name
Parámetro en un subprograma	p_param_name
Cursor	cur_cursor_name
Excepción	e_exception_type

Operadores:

Los operadores que emplea un programa PL/SQL comprende a los operadores clásicos que se encuentran en la mayoría de los lenguajes: operadores aritméticos, operadores lógicos, etc. Existen algunas diferencias particulares que se muestran a continuación:

- El operador = se emplea para hacer comparaciones (no se emplea el operador ==)
- El operador de negación puede ser cualquiera de los siguientes: !=, <>, ~=, ^= • El operador para concatenar cadenas es ||

15

Estructuras de control

Sentencia IF-THEN

```
if condición then
    secuencia_de_instrucciones
End if;
```

Sentencia IF-THEN-ELSE

```
if condición then
    secuencia_de_instrucciones1
else
    secuencia_de_instrucciones2
end if;
```



16

Estructuras de control



Sentencia IF-THEN-ELSIF

```
if condición then
    secuencia_de_instrucciones1
elsif condición2 then
    secuencia_de_instrucciones2
else
    secuencia_de_instrucciones3
end if;
```

17

Sentencia CASE

Al igual que IF, la sentencia CASE selecciona una secuencia de sentencias a ejecutar. Existen 2 estilos:

- La instrucción case contiene un valor escalar (simple case)
- La sentencia case no contiene expresión o valor, se emplean las expresiones booleanas definidas en la instrucción when.

```
case [selector]
    when expresión1 then
        secuencia_de_instrucciones1;
    when expresión2 then
        secuencia_de_instrucciones2;
    ...
    when expresiónn then
        secuencia_de_instruccionesn;
    [else secuencia_de_instrucciones n+1];
end case;
```

18



Ejemplo: Estilo 1: Uso de un escalon.

```
declare
    v_suerte number := 3;
begin
    case v_suerte
    when 1 then
        dbms_output.put_line('El numero de la suerte es UNO');
    when 2 then
        dbms_output.put_line('El numero de la suerte es DOS');
    when 3 then
        dbms_output.put_line('El numero de la suerte es TRES');
    else
        dbms_output.put_line('Numero de la suerte invalido');
    end case;
end;
/
```



19

Ejemplo: Estilo 2: Uso de una expresión booleana en la instrucción when.

```
declare
    v_suerte number := 3;
begin
    case
    when v_suerte = 1 then
        dbms_output.put_line('El numero de la suerte es UNO');
    when v_suerte = 2 then
        dbms_output.put_line('El numero de la suerte es DOS');
    when v_suerte = 3 then
        dbms_output.put_line('El numero de la suerte es TRES');
    else
        dbms_output.put_line('Numero de la suerte invalido');
    end case;
end;
/
```



20

Estructuras de Iteración:



PL/SQL soporta 3 estructuras de iteración:

- For Loops
- Simple Loops
- While Loops

For Loops:

Generalmente un For loop se emplea para recorrer cursores, pero también se pueden emplear para otros propósitos empleado un For numérico (iteración por rango de valores).

Sintaxis:

```
for i in starting_number .. ending_number loop
    <statements>
end loop;
```

21

Ejemplo:



```
begin
  for v_index in 1 .. 10 loop
    dbms_output.put_line('Iterando : ' || v_index);
    if v_index >= 5 then
      dbms_output.put_line('terminando prematuramente');
      exit;
    end if;
    if v_index = 2 then
      dbms_output.put_line('El siguiente numero sera el 3');
      continue;
    end if;
  end loop;
end;
/
```

22

While Loops:

Sintaxis:

```
while <condicion> loop
    statement 1;
    statement 2;
    ...
end loop;
```

Como se puede observar, su estructura es muy similar al clásico **while** encontrado en otros lenguajes. Puede ser empleado con cursores, pero en la práctica se prefiere el uso de un **for loop**.

23



Ejemplo:

```
declare
    v_index number := 1;

begin
    while v_index <= 10 loop
        dbms_output.put_line(v_index);
        v_index := v_index+1;
    end loop;

end;
/
```

24



PROCEDIMIENTOS ALMACENADO



- El lenguaje que se emplea para programar varía de un Sistema Manejador de Bases de Datos Relacional (RDBMS) a otro. No existe un estándar como en el caso del SQL.
- El lenguaje que utiliza Oracle se llama PL/SQL (Procedural Lenguaje extensión to SQL) y es un lenguaje de programación que se usa para acceder y trabajar con bases de datos en Oracle desde distintos entornos.

25

25

TIPOS DE PROCEDIMIENTOS ALMACENADOS



- A nivel de esquema.
- Dentro de un paquete
- Dentro de un bloque PL/SQL

PARTES DE UN PROCEDIMIENTO ALMACENADOS

- Parte declarativa.
- Parte ejecutable.
- Manejo de excepciones

26

26

TRIGGERS EN PL/SQL



- Es un bloque de código que se ejecuta automáticamente cada vez que ocurre algún evento en la base de datos, ya sea inserción de datos, actualización o borrado de datos de una tabla siempre y cuando se intente modificar y se desee llevar registro de cualquier modificación entonces se realiza la programación de un trigger o disparador para asociarlo al evento que ocurre en un momento dado en la base de datos

27

27

Funcionalidad de un Trigger



- . Conservar la integridad referencial y la coherencia entre los datos entre distintas tablas.
- Registrar los cambios que se efectúan sobre las tablas y la identidad de quien los realizo.
- Realizar cualquier acción cuando una tabla es modificada.
- Insertar, actualizar o eliminar datos de una tabla asociada en forma automática

28

28

Reglas de los Triggers



- No pueden ser invocados directamente.
- Al intentar modificar los datos de una tabla asociada, el trigger se ejecuta automáticamente.
- No reciben ni retornan parametros.
- No generan resultados de consultas SQL

29

29

Clasificación de los triggers



- El momento en que se dispara: si se ejecutan antes (before) o después (after) de la sentencia.
- El evento que los dispara: insert, update o delete, según se ejecute una de estas sentencias sobre la tabla.
- Nivel: dependiendo si se ejecuta para cada fila afectada en la sentencia (por cada fila) o bien una única vez por sentencia independientemente de las filas afectadas (nivel de sentencia).

30

30

Paquetes de PLSQL



Son un grupo lógico de objetos de la Base de Datos tales como:

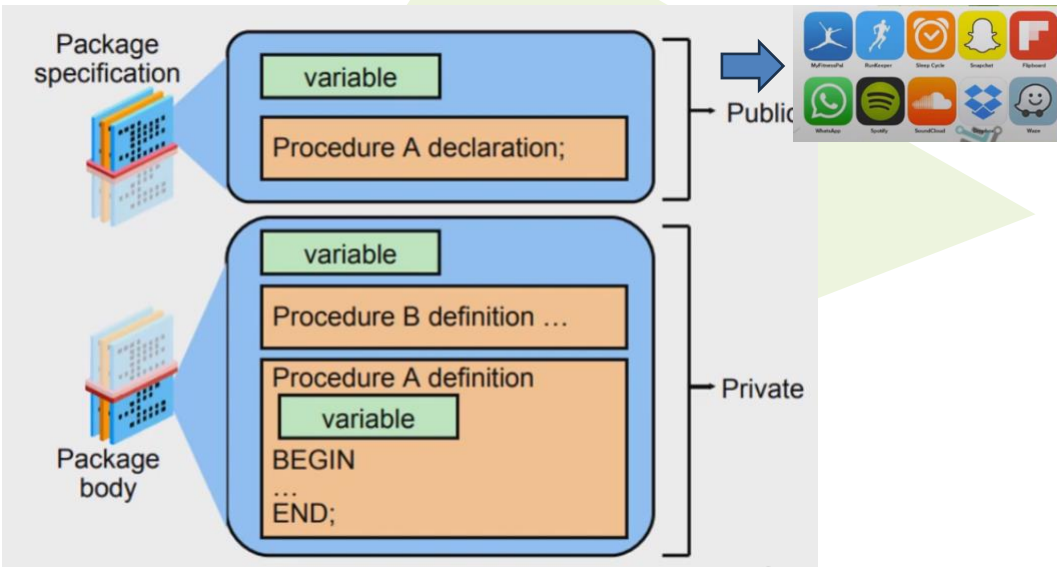
- Registros de datos PLSQL
- Variables
- Estructura de datos
- Excepciones
- Subprogramas
- Procedimientos
- Funciones

Estos objetos se relacionan entre si, encapsulados y convertidos en una unidad dentro de la Base de Datos

31

31

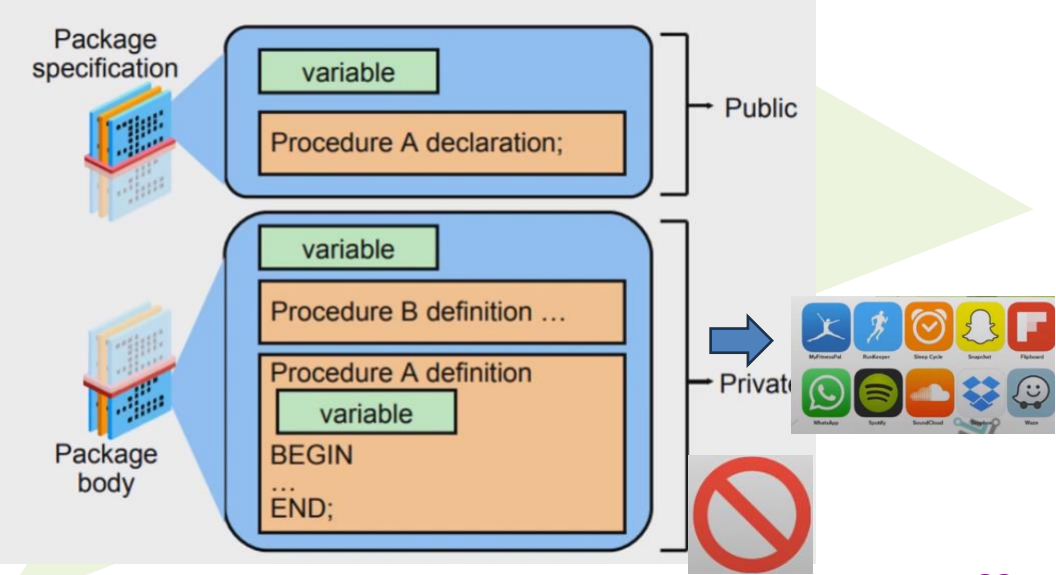
Composición de un Paquete:



32

32

Composición de un Paquete:



33

33

¿Preguntas?



34