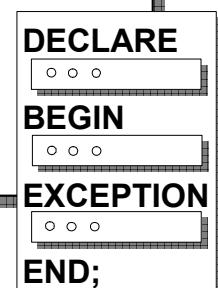


Programación en PL/SQL

ORACLE®

Estructura de un bloque PL/SQL

- **DECLARE** – Opcional
 - Variables, cursores, excepciones definidas por el usuario
- **BEGIN** – Obligatorio
 - Órdenes SQL
 - Órdenes PL/SQL
- **EXCEPTION** – Opcional
 - Acciones a realizar cuando sucede un error
- **END;** – Obligatorio



Estructura de un bloque PL/SQL

```
DECLARE
    variable VARCHAR2 (5);
BEGIN
    SELECT      nombre_columna
                INTO      variable
                FROM      nombre_tabla;
EXCEPTION
    WHEN nombre_expcion THEN
        ...
END;
```

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

PL/SQL-3

ORACLE®

Tipos de Bloques

Anónimo

```
[DECLARE]

BEGIN
    --órdenes

[EXCEPTION]

END ;
```

Procedimiento

```
PROCEDURE nombre
IS

BEGIN
    --órdenes

[EXCEPTION]

END ;
```

Función

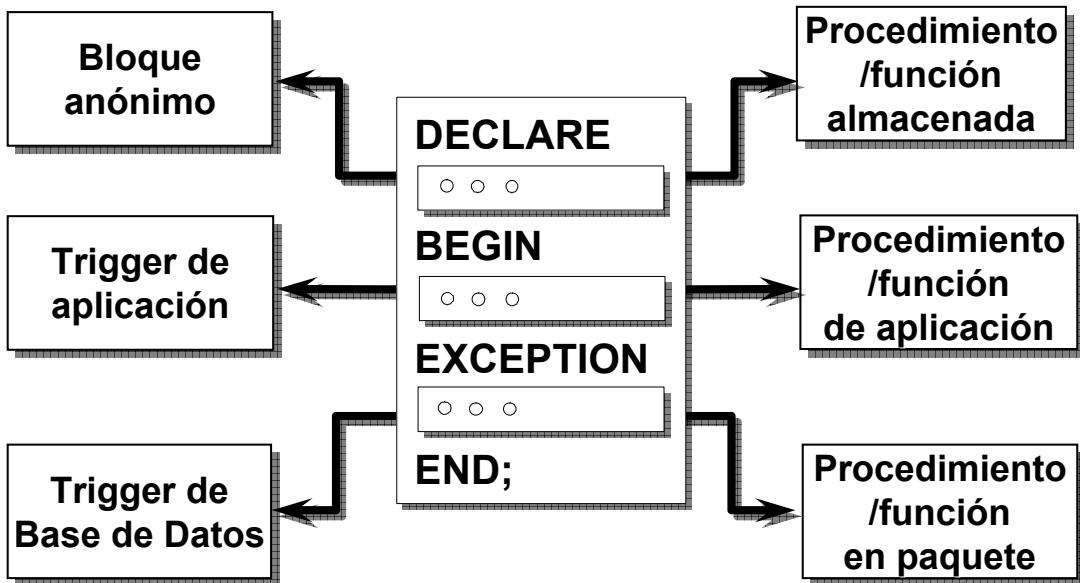
```
FUNCTION nombre
RETURN tipo
IS
BEGIN
    --órdenes
    RETURN valor;
[EXCEPTION]

END ;
```

PL/SQL-4

ORACLE®

Aplicaciones



PL/SQL-5

ORACLE®

Tipos de Variables

- **Variables PL/SQL**
 - Escalar
 - Compuesta
 - Referencia
 - LOB (large objects)
- **Variables no-PL/SQL**
- **Variables host y acotadas (bind)**

PL/SQL-6

ORACLE®

Declaración de Variables PL/SQL

Sintaxis

```
identificador [CONSTANT] tipo_datos [NOT NULL]  
[ := | DEFAULT expr] ;
```

Ejemplos

```
Declare  
  v_fecha          DATE ;  
  v_nodepto        NUMBER(2) NOT NULL := 10 ;  
  v_ciudad         VARCHAR2(13) := 'Atlanta' ;  
  c_cpostal        CONSTANT NUMBER := 1400 ;
```

PL/SQL-7

ORACLE®

Asignación de Valores a Variables

Sintaxis

```
identifier := expr ;
```

Ejemplos

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

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

PL/SQL-8

ORACLE®

Tipos de datos escalares

- **VARCHAR2 (longitud_máxima)**
- **NUMBER [(precisión, escala)]**
- **DATE**
- **CHAR [(longitud_máxima)]**
- **LONG**
- **LONG RAW**
- **BOOLEAN**
- **BINARY_INTEGER**
- **PLS_INTEGER**

PL/SQL-9

ORACLE®

Declaraciones de Variables escalares

Ejemplos

```
v_trabajo      VARCHAR2(9);
v_contador    BINARY_INTEGER := 0;
v_salario     NUMBER(9,2)  := 0;
v_fecha       DATE := SYSDATE + 7;
c_iva          CONSTANT NUMBER(3,2) := 8.25;
v_valido      BOOLEAN NOT NULL := TRUE;
```

PL/SQL-10

ORACLE®

Atributo %TYPE

- **Sufijo que declara una variable coincidente con:**
 - Una definición de una columna de la base de datos
 - Otra variable previamente declarada

```
...
v_nombre          emp.nombre%TYPE;
v_balance         NUMBER(7,2);
v_balance_min    v_balance%TYPE := 10;
...
```

PL/SQL-11

ORACLE®

Tipos de datos compuestos

Tipos

- TABLAS PL/SQL
- REGISTROS PL/SQL

PL/SQL-12

ORACLE®

Referenciando variables No PL/SQL

Almacenar el salario anual en una variable host de SQL*Plus.

```
:g_sal_mensual := v_sal / 12;
```

- Referencia variables no-PL/SQL como variables host.
- Prefijar las referencias con (:).

PL/SQL-13

ORACLE®

Comentarios

- Una línea con dos guiones (- -).
- Múltiples líneas entre los símbolos /* y */.

Ejemplo

```
...
  v_sal NUMBER (9,2);
BEGIN
  /* Calcula el salario anual basado en el salario
  mensual introducido por el usuario */
  v_sal := v_sal * 12;
END; -- Final de la transacción
```

PL/SQL-14

ORACLE®

Funciones SQL en PL/SQL

- Disponibles:
 - Numéricas una-fila
 - De caracteres una-fila
 - Conversión de tipos
 - Fecha
 - No disponibles:
 - GREATEST
 - LEAST
 - DECODE
 - Funciones de grupo
- 
- Igual en SQL

PL/SQL-15

ORACLE®

Funciones PL/SQL

Ejemplos

- Construir una lista de correo.

```
v_direccion_correo := v_nombre||CHR(10)||  
                      v_direccion||CHR(10)||v_ciudad||  
                      CHR(10)||v_codigopostal;
```

- Conversión a minúsculas.

```
v_nombre      := LOWER(v_nombre);
```

PL/SQL-16

ORACLE®

Funciones de conversión

- **TO_CHAR**
- **TO_DATE**
- **TO_NUMBER**

```
BEGIN
    SELECT TO_CHAR(fecha,
                  'MON. DD, YYYY')
    FROM   emp;
END;
```

```
v_comment := USER||': '||TO_CHAR(SYSDATE);
```

PL/SQL-17

ORACLE®

Bloques anidados y Variables

- Los bloques pueden anidarse.

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

Alcance de x

Alcance de y

PL/SQL-18

ORACLE®

Operadores en PL/SQL

- Lógicos
- Aritméticos
- Concatenación
- Paréntesis
- Operador exponencial (**)

}

Igual
en SQL

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

PL/SQL-19

ORACLE®

Variables Bind

Para referenciar una variable bind debe prefijarse con :.

Ejemplo

```
DECLARE
    v_sal      emp.sal%TYPE;
BEGIN
    SELECT    sal
    INTO      v_sal
    FROM      emp
    WHERE     empno = 7369;
    :salary   := v_sal;
END;
```

PL/SQL-20

ORACLE®

Interaccionando con el servidor Oracle

ORACLE®

Órdenes SQL en PL/SQL

- Orden **SELECT** . Sólo puede obtenerse una tupla (cláusula **INTO**).
- Órdenes **DML**.
- Control de transacciones:
COMMIT, **ROLLBACK**, o **SAVEPOINT**

```
SELECT lista_select
INTO   {nombre_variable[, nombre_variable]...
       | nombre_registro}
FROM   tabla
WHERE  condición;
```

Órdenes SQL en PL/SQL

```
DECLARE
    v_deptno      NUMBER(2);
    v_loc         VARCHAR2(15);
BEGIN
    SELECT      deptno, loc
    INTO        v_deptno, v_loc
    FROM        dept
    WHERE       dname = 'VENTAS';
    ...
END;
```

```
DECLARE
    v_empno      emp.empno%TYPE;
BEGIN
    SELECT      secuencia_empno.NEXTVAL
    INTO        v_empno
    FROM        dual;
    INSERT INTO emp(empno, nombre, trabajo, deptno)
    VALUES(v_empno, 'PEPE', 'AUXILIAR', 10);
END;
```

PL/SQL-23

ORACLE®

Órdenes COMMIT y ROLLBACK

- Una transacción se inicia con la primera orden DML command que sigue a COMMIT o ROLLBACK.
- Usar las órdenes SQL COMMIT y ROLLBACK para terminar explícitamente una transacción

PL/SQL-24

ORACLE®

Cursor SQL

- Un cursor es un área privada de trabajo SQL.
- Hay dos tipos de cursos:
 - Implícitos
 - Explícitos
- El servidor Oracle usa cursos implícitos para analizar y ejecutar las órdenes SQL.
- Los cursos explícitos se declaran por el programador explícitamente.

Atributos de Cursor

Se utilizan para chequear la salida de las órdenes SQL.

SQL%ROWCOUNT	Número de filas afectadas por la orden SQL más reciente (valor de tipo integer).
SQL%FOUND	Atributo booleano que se evalúa a TRUE si la orden SQL más reciente afecta a una o más filas.
SQL%NOTFOUND	Atributo booleano que se evalúa a TRUE si la orden SQL más reciente no afecta a ninguna fila.
SQL%ISOPEN	Siempre evaluado a FALSE porque PL/SQL cierra los cursos implícitos después de la ejecución.

Atributos de Cursor

**Borra las filas de la tabla ITEM que tienen el número de orden especificado.
Imprime el número de filas borradas.**

Ejemplo

```
VARIABLE filas_borradas
DECLARE
    v_ordid NUMBER := 605;
BEGIN
    DELETE FROM item
    WHERE ordid = v_ordid;
    filas_borradas := SQL%ROWCOUNT
        ||' filas borradas.');
END;
PRINT filas_borradas
```

PL/SQL-27

ORACLE®

Estructuras de Control

ORACLE®

Órdenes IF

Sintaxis

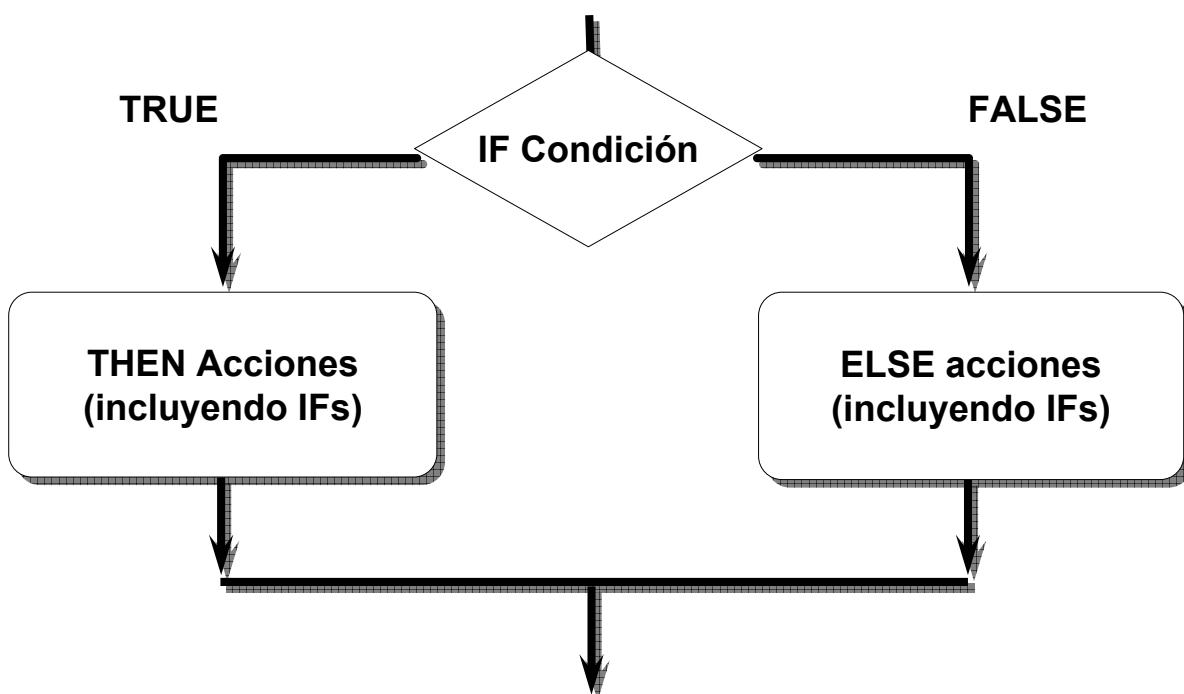
```
IF condición THEN  
    órdenes;  
[ELSIF condición THEN  
    órdenes;]  
[ELSE  
    órdenes;]  
END IF;
```

```
IF v_ename = 'OSBORNE' THEN  
    v_mgr := 22;  
END IF;
```

PL/SQL-29

ORACLE®

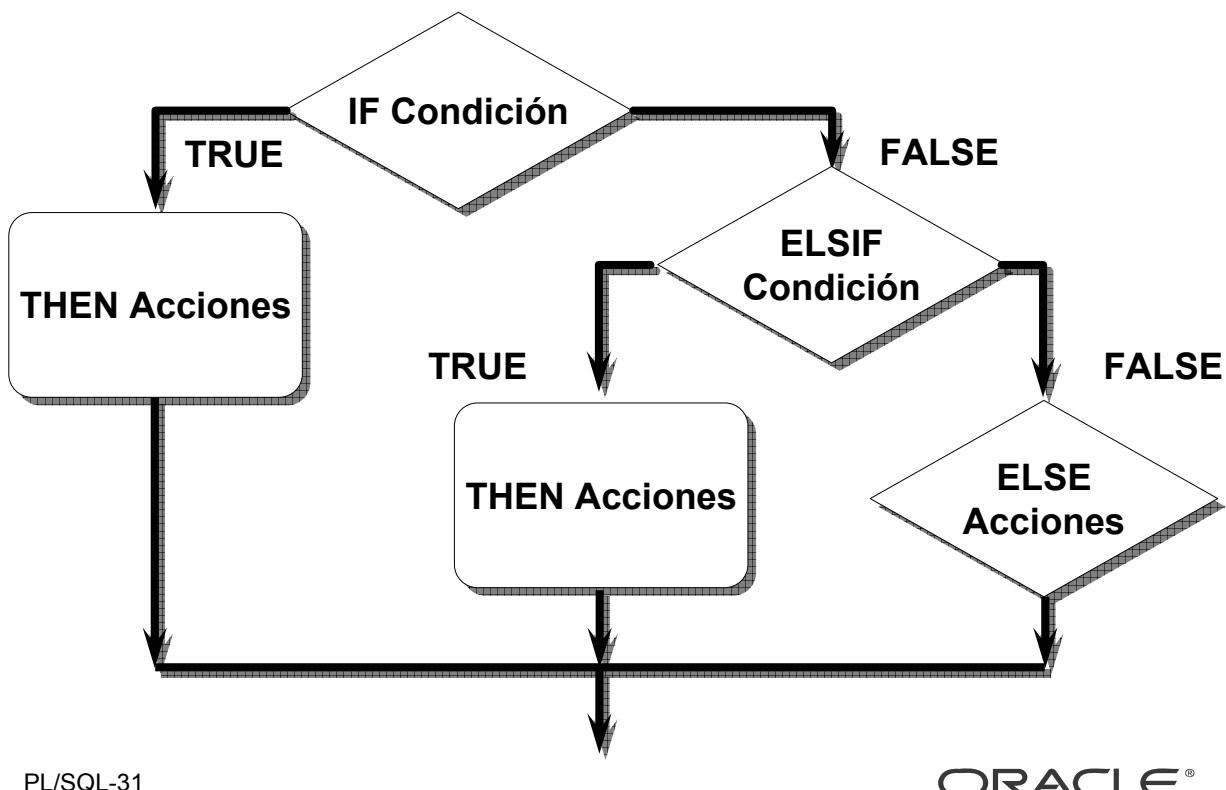
Flujo IF-THEN-ELSE



PL/SQL-30

ORACLE®

Flujo IF-THEN-ELSIF



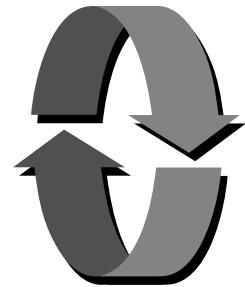
Valores nulos

- Se pueden gestionar valores nulos con el operador IS NULL.
- Cualquier expresión que contenga un valor nulo se evalúa a NULL.
- Las expresiones que se concatenan con valores nulos los tratan como cadenas vacías.

Control de iteraciones

Hay tres tipos de bucles:

- Bucle básico
- Bucle FOR
- Bucle WHILE



PL/SQL-33

ORACLE®

Bucle básico

Sintaxis

```
LOOP                                -- delimitador
  orden1;                            -- órdenes
  .
  .
  EXIT [WHEN condición];          -- orden EXIT
END LOOP;                            -- delimitador
```

donde: *condición* es una variable booleana o
expresión (TRUE, FALSE,
o NULL);

PL/SQL-34

ORACLE®

Bucle básico

Ejemplo

```
DECLARE
    v_ordid      item.ordid%TYPE := 101;
    v_contador  NUMBER(2)  := 1;
BEGIN
    LOOP
        INSERT INTO item(ordid, itemid)
            VALUES(v_ordid, v_contador);
        v_contador := v_contador + 1;
        EXIT WHEN v_contador > 10;
    END LOOP;
END;
```

PL/SQL-35

ORACLE®

Bucle FOR

Sintaxis

```
FOR contador in [REVERSE]
    cota_inerior..cota_superior LOOP
    órdenes;
    .
    .
    .
END LOOP;
```

No hay que declarar el índice; se declara implícitamente.

```
DECLARE
    v_ordid      item.ordid%TYPE := 101;
BEGIN
    FOR i IN 1..10 LOOP
        INSERT INTO item(ordid, itemid)
            VALUES(v_ordid, i);
    END LOOP;
END;
```

PL/SQL-36

ORACLE®

Bucle WHILE

Sintaxis

```
WHILE condicion LOOP
    orden1;
    orden2;
    . . .
END LOOP;
```

La condición
se evalúa al
Comienzo de
cada iteración.

PL/SQL-37

ORACLE®

Bucle WHILE

Ejemplo

```
ACCEPT p_precio PROMPT 'Introduce el precio: '
ACCEPT p_itemtot PROMPT 'Introduce el máximo: '
DECLARE
    ...
    v_ctd      NUMBER(8) := 1;
    v_total    NUMBER(7,2) := 0;
BEGIN
    ...
    WHILE v_total < &p_itemtot LOOP
        ...
        v_ctd := v_ctd + 1;
        v_total := v_ctd * &p_precio;
    END LOOP;
    ...

```

PL/SQL-38

ORACLE®

Bucles anidados y etiquetas

- Se pueden anidar bucles en múltiples niveles.
- Se pueden utilizar etiquetas para distinguir entre bucles y bloques.
- Se puede salir del bucle externo con la orden EXIT referenciando la etiqueta.

Bucles anidados y etiquetas

```
...
BEGIN
  <<bucle_externo>>
  LOOP
    v_contador := v_contador+1;
    EXIT WHEN v_contador>10;
    <<bucle_interno>>
    LOOP
      ...
      EXIT bucle_externo WHEN total_realizado = 'SI';
      -- Deja ambos bucles
      EXIT WHEN interno_realizado = 'SI';
      -- Deja sólo el bucle interno
      ...
    END LOOP bucle_interno;
  ...
END LOOP bucle_externo;
END;
```

Tipos de Datos Compuestos

ORACLE®

Tipos de datos compuestos

- **Tipos:**
 - **REGISTROS PL/SQL**
 - **TABLAS PL/SQL**
- **Contienen componentes internos**
- **Son reutilizables**

Registro PL/SQL

Sintaxis

```
TYPE nombre_tipo IS RECORD  
    (declaración_campo[, declaración_campo]...);  
identificador nombre_tipo;
```

Donde *declaración_campo* es:

```
nombre_campo {tipo_campoe | variable%TYPE  
| tabla.columna%TYPE | tabla%ROWTYPE}  
[[NOT NULL] {:= | DEFAULT} expr]
```

Registro PL/SQL

Ejemplo

```
...  
TYPE empleado_tipo_reg IS RECORD  
    (nombre      VARCHAR2(10),  
     trabajo     VARCHAR2(9),  
     salario     NUMBER(7,2));  
emp_registro  empleado_tipo_reg;  
...
```

Atributo %ROWTYPE

- Declara una variable de acuerdo a una colección de columnas en una tabla de base de datos o vista.
- Prefija %ROWTYPE con la tabla.
- Los campos del registro toman nombres y tipos de las columnas de la tabla o vista.

```
dept_record    dept%ROWTYPE;
```

```
emp_record    emp%ROWTYPE;
```

PL/SQL-45

ORACLE®

Tablas PL/SQL

- Están constituidas por dos componentes:
 - Clave Primaria de tipo **BINARY_INTEGER**
 - Columnas de tipo escalar or registro
- Crece dinámicamente porque no está limitada

PL/SQL-46

ORACLE®

Tablas PL/SQL

Sintaxis

```
TYPE nombre_tipo IS TABLE OF
  {tipo_columna | variable%TYPE
  | tabla.columna%TYPE} [NOT NULL]
  [INDEX BY BINARY_INTEGER];
Identificador          nombre_tipo;
```

Ejemplo

```
...
TYPE nombre_tipo_tabla IS TABLE OF emp.nombre%TYPE
  INDEX BY BINARY_INTEGER;
nombre_tabla nombre_tipo_tabla;
...
```

PL/SQL-47

ORACLE®

Estructura de tabla PL/SQL

Clave Primaria

...
1
2
3
...

Columna

...
Luís
Pepe
Manolo
...

BINARY_INTEGER

Escalar

PL/SQL-48

ORACLE®

Tabla PL/SQL

```
DECLARE
    TYPE nombre_tipo_tabla IS TABLE OF emp.nombre%TYPE
        INDEX BY BINARY_INTEGER;
    TYPE fecha_tipo_tabla IS TABLE OF DATE
        INDEX BY BINARY_INTEGER;
    nombre_tabla      nombre_tipo_tabla;
    fecha_tabla       fecha_tipo_tabla;
BEGIN
    nombre_tabla(1) := 'CARMELO';
    fecha_tabla(8) := SYSDATE + 7;
    IF nombre_tabla.EXISTS(1) THEN
        INSERT INTO ...
    ...
END;
```

PL/SQL-49

ORACLE®

Tabla de Registros PL/SQL

- Define a variable TABLE con el atributo %ROWTYPE.
- Declara una variable PL/SQL para almacenar información del departamento.

Ejemplo

```
DECLARE
    TYPE dept_tipo_tabla IS TABLE OF dept%ROWTYPE
        INDEX BY BINARY_INTEGER;
    dept_tabla dept_tipo_tabla;
    -- Cada elemento de dept_tabla es un registro
```

PL/SQL-50

ORACLE®

Métodos de Tabla PL/SQL

Para facilitar el uso de la tabla PL/SQL:

- EXISTS
- COUNT
- FIRST and LAST
- PRIOR
- NEXT
- EXTEND
- TRIM
- DELETE

PL/SQL-51

ORACLE®

Cursos Explícitos

ORACLE®

Cursos

Cada orden SQL ejecutada por el servidor Oracle tiene asociada un cursor individual:

- **Cursos implícitos:** Declarados para todas las órdenes DML y órdenes SELECT PL/SQL.
- **Cursos explícitos:** Declarados y nominados por el programador.

Cursor Explícito

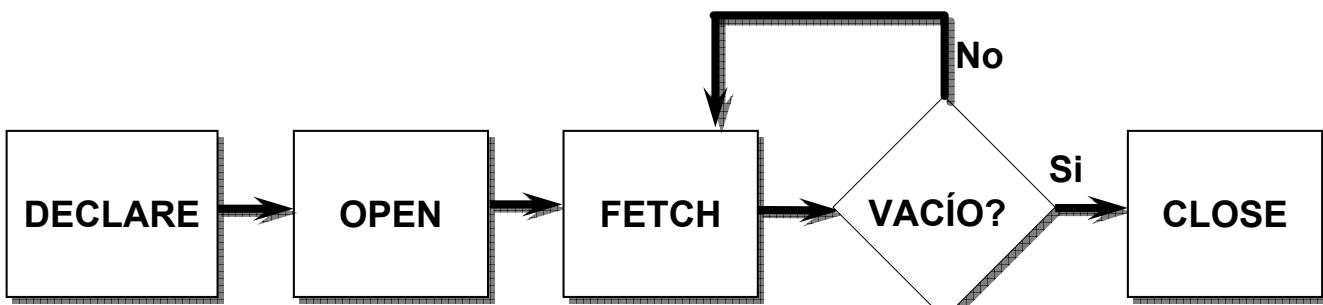
Conjunto resultado

Cursor

7369	SMITH	CLERK
7566	JONES	MANAGER
7788	SCOTT	ANALYST
7876	ADAMS	CLERK
7902	FORD	ANALYST

Fila actual

Control de cursos explícitos



- Crea un área SQL nominada
- Identifica el conjunto activo
- Almacena la fila actual en variables
- Chequea si existen filas
- Retorna a FETCH si encuentra filas
- Elimina el conjunto activo

PL/SQL-55

ORACLE®

Declarando el Cursor

Sintaxis

```
CURSOR nombre_cursor IS  
    orden_select;
```

```
DECLARE  
    CURSOR c1 IS  
        SELECT empno, nombre  
        FROM emp;  
  
    CURSOR c2 IS  
        SELECT *  
        FROM dept  
        WHERE deptno = 10;  
BEGIN  
    ...
```

PL/SQL-56

ORACLE®

Abriendo el Cursor

Sintaxis

```
OPEN nombre_cursor;
```

Obteniendo Datos del Cursor

```
FETCH nombre_cursor INTO [variable1, variable2, ...]  
| nombre_registro;
```

Cerrando el Cursor

```
CLOSE nombre_cursor;
```

PL/SQL-57

ORACLE®

Uso del Cursor

Ejemplo

```
FETCH c1 INTO v_empno, v_nombre;
```

```
...  
OPEN cursor1;  
LOOP  
  FETCH cursor1 INTO variables  
  EXIT WHEN ...;  
  ...  
    -- Procesa los datos obtenidos  
  ...  
END;  
CLOSE cursor1;
```

PL/SQL-58

ORACLE®

Atributos de Cursor Explícito

Obtiene información sobre el estado del cursor.

Atributo	Tipo	Descripción
%ISOPEN	Boolean	Evalúa a TRUE si el cursor está abierto
%NOTFOUND	Boolean	Evalúa a TRUE si el fetch más reciente no retorna filas
%FOUND	Boolean	Evalúa a TRUE si el fetch más reciente retorna filas; complemento de %NOTFOUND
%ROWCOUNT	Number	Evalúa al número total de filas retornadas

PL/SQL-59

ORACLE®

Ejemplos

```
IF NOT c1%ISOPEN THEN
    OPEN c1;
END IF;
LOOP
    FETCH c1...
```

```
...
CURSOR c1 IS
    SELECT empno, nombre
    FROM emp;
    emp_registro c1%ROWTYPE;
BEGIN
    OPEN c1;
    . . .
    FETCH c1 INTO emp_registro;
```

PL/SQL-60

ORACLE®

Bucle FOR Cursor

Sintaxis

```
FOR nombre_registro IN nobre_cursor LOOP  
    orden1;  
    orden2;  
    . . .  
END LOOP;
```

- Acorta el proceso para cursores explícitos.
- Incluye implicitamente open, fetch, y close.
- El registro se declara implícitamente.

PL/SQL-61

ORACLE®

Bucle FOR Cursor

Ejemplo

```
DECLARE  
    CURSOR c1 IS  
        SELECT empno, nombre  
        FROM emp;  
BEGIN  
    FOR emp_registro IN c1 LOOP  
        -- se ejecuta open y fetch implicitamente  
        IF emp_registro.empno = 7839 THEN  
            . . .  
        END LOOP; -- se ejecuta close implicitamente  
    END;
```

PL/SQL-62

ORACLE®

Bucle FOR Cursor Usando Subconsultas

No es necesario declarar el cursor.

Ejemplo

```
BEGIN
    FOR emp_registro IN ( SELECT empno, nombre
                           FROM   emp) LOOP
        -- se ejecuta open y fetch implícitamente
        IF emp_registro.empno = 7839 THEN
            ...
        END LOOP; -- se ejecuta close implícitamente
END;
```

PL/SQL-63

ORACLE®

Cursos con Parámetros

Sintaxis

```
CURSOR nombre_cursor
  [ (nombre_parámetro tipo_de_datos, ...) ]
IS
  orden_select;
```

- Pasa parámetros al cursor cuando se abre y la consulta se ejecuta.
- Abre un cursor explícito varias veces con un conjunto activo distinto cada vez.

PL/SQL-64

ORACLE®

Cursos con Parámetros

Ejemplo

```
DECLARE
  CURSOR c1
  (v_deptno NUMBER, v_trabajo VARCHAR2) IS
    SELECT empno, nombre
    FROM emp
    WHERE deptno = v_deptno
      AND trabajo = v_trabajo;
BEGIN
  OPEN c1(10, 'AUXILIAR');
  ...

```

PL/SQL-65

ORACLE®

Cláusula FOR UPDATE

Sintaxis

```
SELECT ...
FROM ...
FOR UPDATE [OF referencia_columna] [NOWAIT]
```

- **Bloquea explícitamente durante la transacción.**
- **Bloquea las filas antes de actualizar o borrar.**

PL/SQL-66

ORACLE®

Cláusula FOR UPDATE

Ejemplo

```
DECLARE
  CURSOR c1 IS
    SELECT empno, nombre
    FROM   emp
    FOR UPDATE NOWAIT;
```

PL/SQL-67

ORACLE®

Cláusula WHERE CURRENT OF

Sintaxis

```
WHERE CURRENT OF cursor
```

- **Usa cursos para actualizar o borrar la fila actual.**
- **Incluir la cláusula FOR UPDATE en la consulta del cursor para primero bloquear las filas.**
- **Usar la cláusula WHERE CURRENT OF para referenciar la fila actual de un cursor explícito.**

PL/SQL-68

ORACLE®

Cláusula WHERE CURRENT OF

Ejemplo

```
DECLARE
    CURSOR c1 IS
        SELECT ...
        FOR UPDATE NOWAIT;
BEGIN
    ...
    FOR emp_registro IN c1 LOOP
        UPDATE ...
            WHERE CURRENT OF c1;
    ...
END LOOP;
COMMIT;
END;
```

PL/SQL-69

ORACLE®

Cursos con Subconsultas

Ejemplo

```
DECLARE
    CURSOR cursor1 IS
        SELECT t1.deptno, nombre, PLANTILLA
        FROM dept t1, (SELECT deptno,
                            count(*) PLANTILLA
                            FROM emp
                            GROUP BY deptno) t2
        WHERE t1.deptno = t2.deptno
        AND PLANTILLA >= 5;
```

PL/SQL-70

ORACLE®

Excepciones

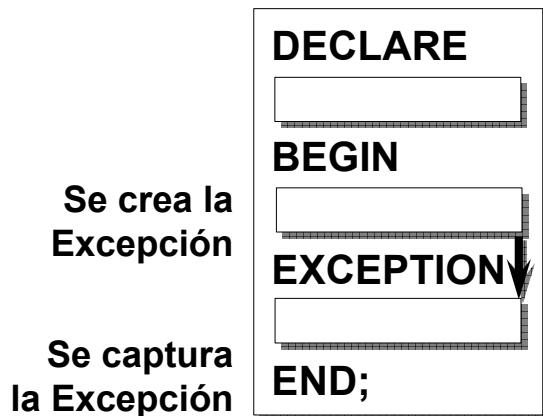
ORACLE®

Gestionando Excepciones con PL/SQL

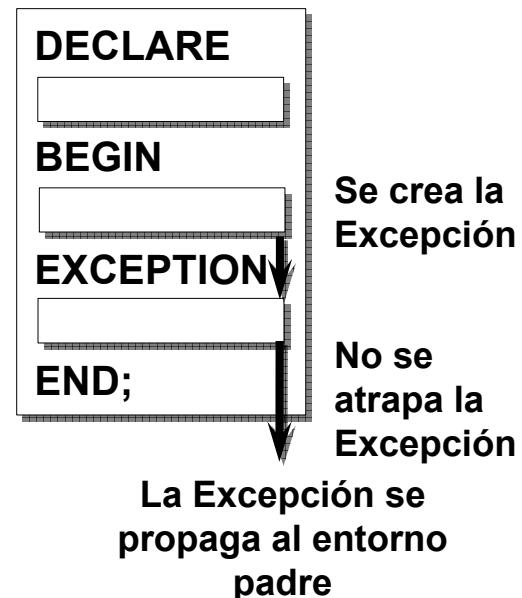
- **¿Qué es una excepcion?**
 - Identificador en PL/SQL que surge durante la ejecución.
- **¿Cómo surge?**
 - Cuando ocurre un error Oracle.
 - O se crea explícitamente.
- **¿Cómo gestionarlo?**
 - Capturarlo con un manejador.
 - Propagarlo al entorno padre.

Gestionando Excepciones

Captura la Excepción



Propaga la Excepción



PL/SQL-73

ORACLE®

Tipos de Excepción

- Predefinida por el servidor Oracle
 - No predefinida por el servidor Oracle
 - Definida por el usuario
- } Surge Implícitamente
- } Surge Explícitamente

PL/SQL-74

ORACLE®

Capturando Excepciones

Sintaxis

```
EXCEPTION
  WHEN excepción1 [OR excepción2 . . .] THEN
    orden1;
    orden2;
    . . .
  [WHEN excepción3 [OR excepción4 . . .] THEN
    orden1;
    orden2;
    . . .]
  [WHEN OTHERS THEN
    orden1;
    orden2;
    . . .]
```

PL/SQL-75

ORACLE®

Capturando errores predefinidos del servidor Oracle

- Referenciar el nombre estandard en la rutina de gestión de la excepción.
- Ejemplos de excepciones predefinidas:
 - **NO_DATA_FOUND**
 - **TOO_MANY_ROWS**
 - **INVALID_CURSOR**
 - **ZERO_DIVIDE**
 - **DUP_VAL_ON_INDEX**

PL/SQL-76

ORACLE®

Excepción Predefinida

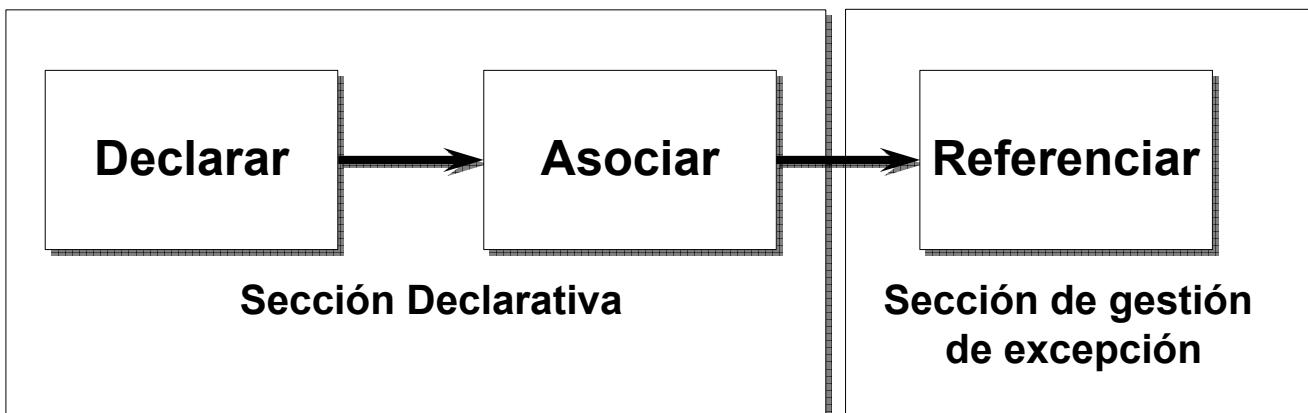
Sintaxis

```
BEGIN  SELECT ... COMMIT;  
EXCEPTION  
    WHEN NO_DATA_FOUND THEN  
        orden1;  
        orden2;  
    WHEN TOO_MANY_ROWS THEN  
        orden1;  
    WHEN OTHERS THEN  
        orden1;  
        orden2;  
        orden3;  
END;
```

PL/SQL-77

ORACLE®

Capturando errores del servidor Oracle no predefinidos



- Nombrar la excepción
- Codificar el PRAGMA EXCEPTION_INIT
- Gestionar la excepción surgida

PL/SQL-78

ORACLE®

Error No-Predefinido

Captura del error de Oracle número -2292

una violación de restricción de integridad

```
DECLARE
    e_producto_invalido      EXCEPTION;
    PRAGMA EXCEPTION_INIT (
        e_producto_invalido, -2292);

    v_mensaje VARCHAR2(50);
BEGIN
    . . .
EXCEPTION
    WHEN e_producto_invalido THEN
        :g_mensaje := 'El código del producto
                       especificado no es válido.';
    . . .
END;
```

1

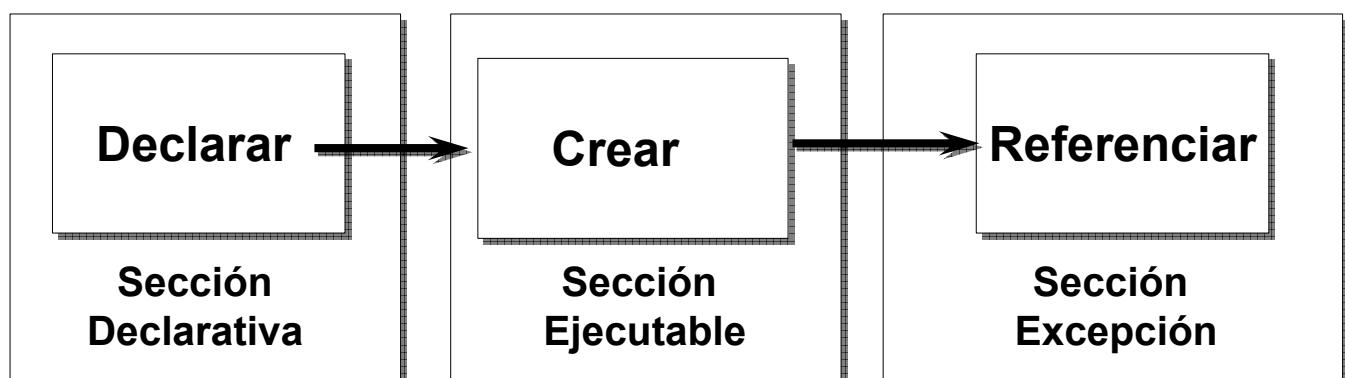
2

3

PL/SQL-79

ORACLE®

Capturando Excepciones definidas por el usuario



- Nombrar la excepción
- Generar explícitamente la excepción usando la orden RAISE
- Gestionar la excepción

PL/SQL-80

ORACLE®

Excepción definida por el usuario

Ejemplo

```
[DECLARE]
    e_cantidad_restante EXCEPTION;
    .
    .
BEGIN
    .
    .
    RAISE e_cantidad_restante;
    .
    .
EXCEPTION
    WHEN e_cantidad_restante THEN
        :g_mensaje := 'Hay stock.';
    .
    .
END;
```

1

2

3

PL/SQL-81

ORACLE®

Funciones para capturar Excepciones

- **SQLCODE**

Retorna el valor numérico del código de error

- **SQLERRM**

Retorna el mensaje asociado con el número de error

PL/SQL-82

ORACLE®

Funciones para capturar Excepciones

Ejemplo

```
DECLARE
    v_codigo_error      NUMBER;
    v_mensaje_error    VARCHAR2 (255);
BEGIN
    ...
EXCEPTION
    ...
    WHEN OTHERS THEN
        ROLLBACK;
        v_codigo_error := SQLCODE ;           ←
        v_mensaje_error:= SQLERRM ;          ←
        INSERT INTO errores VALUES(v_codigo_error,
                                     v_mensaje_error);
END;
```

PL/SQL-83

ORACLE®

Propagando Excepciones

Subbloques pueden
gestionar una
excepción o pasarla al
bloque que lo contiene.

```
DECLARE
    ...
    e_no_filas      exception;
    e_integridad    exception;
    PRAGMA EXCEPTION_INIT (e_integridad, -2292);
BEGIN
    FOR c_registro IN emp_cursor LOOP
        BEGIN
            SELECT ...
            UPDATE ...
            IF SQL%NOTFOUND THEN
                RAISE e_no_filas;
            END IF;
            EXCEPTION
                WHEN e_integridad THEN ...
                WHEN e_no_filas THEN ...
        END;
    END LOOP;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN . . .
        WHEN TOO_MANY_ROWS THEN . . .
    END;
```

PL/SQL-84

ORACLE®

RAISE_APPLICATION_ERROR

Sintaxis

```
raise_application_error (numero_error,  
                      mensaje[, {TRUE | FALSE}]);
```

- Procedimiento que permite crear mensajes de error definidos por el usuario desde suprogramas almacenados.
- Llamado sólo desde un subprograma almacenado en ejecución.

PL/SQL-85

ORACLE®

RAISE_APPLICATION_ERROR

- Se usa en dos lugares diferentes:
 - Sección ejecutable
 - Sección Excepción
- Retorna condiciones de error al usuario de una forma consistente con otros errores del servidor Oracle

PL/SQL-86

ORACLE®

Procedimientos

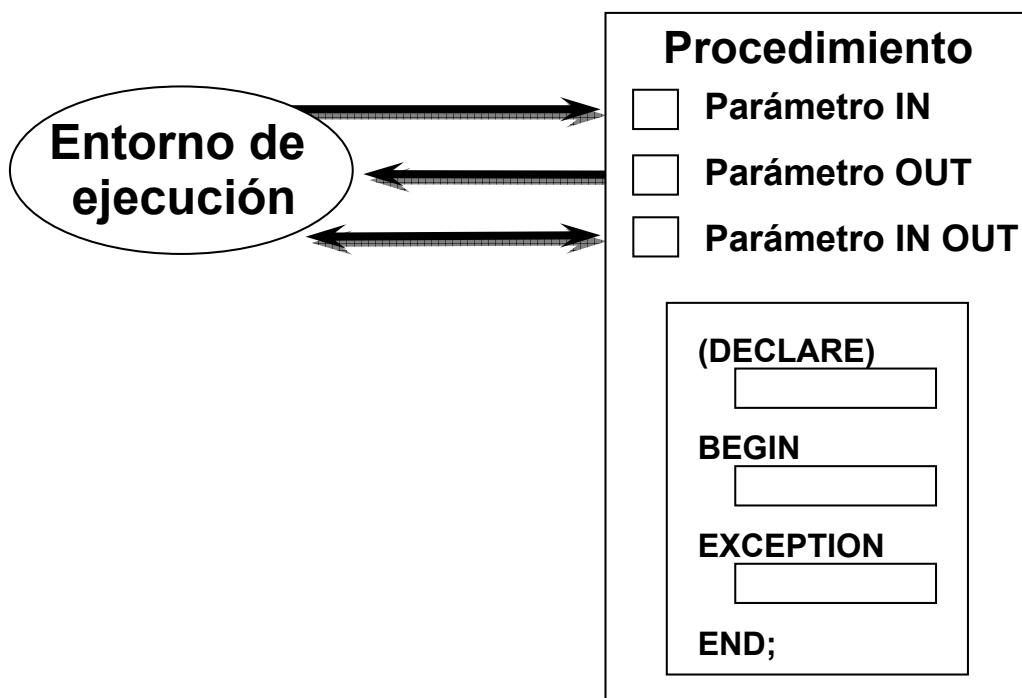
ORACLE®

Procedimientos

- **Un procedimiento es un bloque nominado PL/SQL que realiza una acción.**
- **Puede almacenarse como un objeto de la base de datos, para su ejecución repetida.**

```
CREATE [OR REPLACE] PROCEDURE nombre_procedimiento
  (argumento1 [modo1] tipo_de_datos1,
   argumento2 [modo2] tipo_de_datos2,
   . . .
   IS [AS]
   Bloque PL/SQL;
```

Modos de Parámetros



PL/SQL-89

ORACLE®

Modos de Parámetros

IN	OUT	IN OUT
Por defecto	Debe especificarse	Debe especificarse
El Valor se pasa al subprograma	Retornado al entorno de ejecución	Pasado al subprograma; retornado al entorno de ejecución
El parámetro actúa como constante	variable no inicializada	Variable inicializada
Puede ser: Literal, Expresión, Constante, o Variable inicializada	Debe ser una variable	Debe ser una variable

PL/SQL-90

ORACLE®

Parámetro IN : Ejemplo

```
SQL> CREATE OR REPLACE PROCEDURE act_salario
  2  (v_id in emp.empno%TYPE)
  3  IS
  4  BEGIN
  5      UPDATE emp
  6      SET sal = sal * 1.10
  7      WHERE empno = v_id;
  8  END act_salario;
  9 /
Procedure created.

SQL> EXECUTE act_salario (7369)
PL/SQL procedure successfully completed.
```

PL/SQL-91

ORACLE®

Parámetro OUT : Ejemplo

```
SQL> CREATE OR REPLACE PROCEDURE consulta_emp
  1  (v_id          IN emp.empno%TYPE,
  2   v_nombre      OUTemp.nombre%TYPE,
  3   v_salario     OUTemp.sal%TYPE,
  4   v_comision    OUTemp.com%TYPE)
  5  IS
  6  BEGIN
  7      SELECT nombre, sal, com
  8      INTO   v_nombre, v_salario, v_comision
  9      FROM   emp
 10      WHERE   empno = v_id;
 11  END consulta_emp;
 12 /
```

PL/SQL-92

ORACLE®

Parámetros OUT y SQL*Plus

```
SQL> START consulta_emp.sql
Procedure created.
```

```
SQL> VARIABLE g_nombre      varchar2(15)
SQL> VARIABLE g_salario     number
SQL> VARIABLE g_comision    number
```

```
SQL> EXECUTE consulta_emp (7654,:g_nombre,:g_salario,
2 :g_comision)
PL/SQL procedure successfully completed.
```

```
SQL> PRINT g_nombre
G_NOMBRE
-----
MARTIN
```

PL/SQL-93

ORACLE®

Parámetros OUT y Procedure Builder

```
PL/SQL> .CREATE CHAR g_nombre LENGTH 10
PL/SQL> .CREATE NUMBER g_salario PRECISION 4
PL/SQL> .CREATE NUMBER g_comision PRECISION 4
PL/SQL> ACT_SALARIO (7654, :g_nombre, :g_salario,
+> :g_comision);
PL/SQL> TEXT_IO.PUT_LINE (:g_nombre || ' gana ' ||
+> TO_CHAR(:g_salario) || ' + comisión de ' ||
+> || TO_CHAR(:g_comision));
MARTIN      gana 1250 + comisión de 1400
```

PL/SQL-94

ORACLE®

Parámetros IN OUT

```
SQL> CREATE OR REPLACE PROCEDURE formatea_tlf
  2  (v_tlf IN OUT VARCHAR2)
  3  IS
  4  BEGIN
  5    v_tlf := '(' || SUBSTR(v_tlf,1,3) ||
  6              ')' || SUBSTR(v_tlf,4,3) ||
  7              '-' || SUBSTR(v_tlf,7);
  8  END formatea_tlf;
  9 /
```

```
SQL> VARIABLE g_tlf varchar2(15)

SQL> BEGIN :g_tlf := '8006330575'; END;

SQL> EXECUTE formatea_tlf (:g_tlf)
```

PL/SQL-95

ORACLE®

Métodos de paso de Parámetros

- **Posicional:**

Los valores se listan en el mismo orden que se declaran

- **Nominada:**

Los valores se listan en cualquier orden asociando cada uno con su parámetro usando =>

- **Combinación:**

Lista los primeros valores posicionalmente y el resto de forma nominada

PL/SQL-96

ORACLE®

Paso de Parámetros: Ejemplo

```
SQL> CREATE OR REPLACE PROCEDURE inserta_dept
  1  (v_nombre  IN dept.nombre%TYPE DEFAULT 'desconocido',
  2   v_loc     IN dept.loc%TYPE      DEFAULT 'desconocido')
  3 IS
  4 BEGIN
  5   INSERT INTO dept
  6     VALUES (dept_deptno.NEXTVAL, v_nombre, v_loc);
  7 END inserta_dept;
  8 /
```

```
SQL> begin
  2 add_dept;
  3 add_dept ( 'TRAINING' , 'NEW YORK');
  4 add_dept ( v_loc => 'DALLAS' , v_name =>'EDUCATION');
  5 add_dept ( v_loc => 'BOSTON') ;
  6 end;
  7 /
PL/SQL procedure successfully completed.
```

PL/SQL-97

ORACLE®

Borrando Procedimientos

Usando SQL*Plus:

- Sintaxis

```
DROP PROCEDURE nombre_procedimiento
```

- Ejemplo

```
SQL> DROP PROCEDURE act_salario;
Procedure dropped.
```

PL/SQL-98

ORACLE®

Funciones

ORACLE®

Funciones

- Una función es un bloque nominado PL/SQL que retorna un valor.
- Puede ser almacenado como un objeto de la base de datos.
- Puede ser llamada como parte de una expresión.

```
CREATE [OR REPLACE] FUNCTION nombre_funcion
(argumento1 [modo1] tipo_de_datos1,
 argumento2 [modo2] tipo_de_datos2,
 . . .
 RETURN tipo_de_datos
 IS|AS
 Bloque PL/SQL;
```

Función almacenada: Ejemplos

```
SQL> CREATE OR REPLACE FUNCTION obten_sal
  2  (v_id IN emp.empno%TYPE)
  3  RETURN NUMBER
  4  IS
  5    v_salario    emp.sal%TYPE :=0;
  6  BEGIN
  7    SELECT sal INTO v_salary
  8    FROM   emp WHERE empno = v_id;
  9    RETURN (v_salary);
10 END get_sal;
11 /
```

```
FUNCTION tax (v_value IN NUMBER)
  RETURN NUMBER
IS
BEGIN
  RETURN (v_value * .08);
END tax;
```

PL/SQL-101

ORACLE®

Ejecutando Funciones:Ejemplos

```
SQL> START obten_salario.sql
Procedure created.

SQL> VARIABLE g_salario number
SQL> EXECUTE :g_salario := obten_sal(7934)
PL/SQL procedure successfully completed.

SQL> PRINT g_salario
      G_SALARIO
-----
      1300
```

```
PL/SQL> .CREATE NUMBER x PRECISION 4
PL/SQL> :x := tasa(1000);
PL/SQL> TEXT_IO.PUT_LINE (TO_CHAR(:x));
80
```

PL/SQL-102

ORACLE®

Llamada a Funciones desde expresiones: Restricciones

- Una función definida por el usuario debe ser almacenada, y no de grupo.
- Una función definida por el usuario sólo puede utilizar parámetros IN.
- Tipos de datos CHAR, DATE, or NUMBER, no tipos PL/SQL tales como BOOLEAN, RECORD, or TABLE.
- El tipo de retorno debe ser un tipo interno de Oracle Server.
- No se permiten INSERT, UPDATE, or DELETE.
- No se permiten llamadas a subprogramas que violen las restricciones anteriores.

PL/SQL-103

ORACLE®

Borrando Funciones

Usando SQL*Plus

- Sintaxis

```
DROP FUNCTION nombre_funcion
```

- Ejemplo

```
SQL> DROP FUNCTION obten_salario;  
Function dropped.
```

Paquetes

ORACLE®

Paquetes

- **Grupo lógico que relaciona tipos, items y subprogramas.**
- **Consiste de dos partes:**
 - **Especificación**
 - **Cuerpo**
- **No puede ser llamado, parametrizado o anidado.**
- **Oracle permite leer múltiples objetos en memoria simultáneamente.**

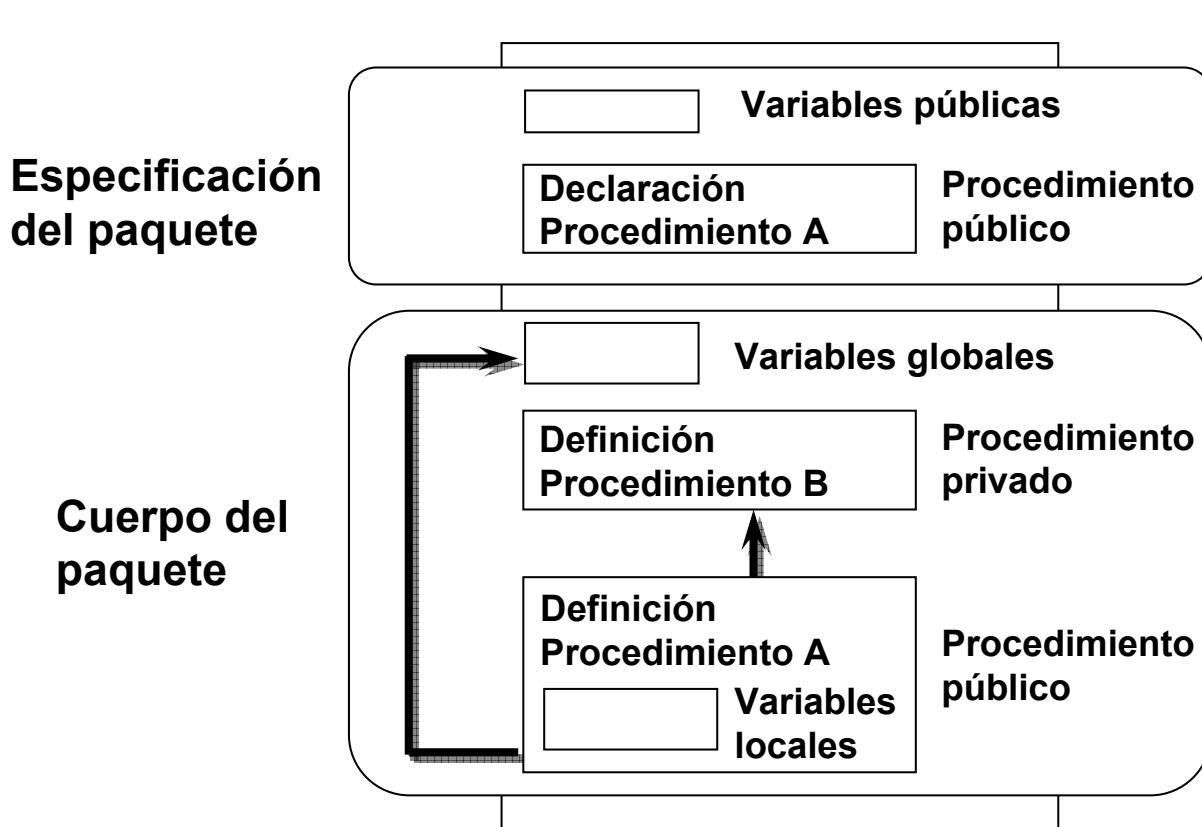
Ventajas de los paquetes

- Modularidad
- Fácil diseño de la aplicación
- Ocultación de la información
- Funcionalidad añadida
- Mejor rendimiento
- Sobrecarga

PL/SQL-107

ORACLE®

Desarrollando un paquete



PL/SQL-108

ORACLE®

Especificación del paquete

Sintaxis

```
CREATE [OR REPLACE] PACKAGE nombre_paquete
IS | AS
    declaracion de items y tipos publicos
    especificaciones de subprogramas
END nombre_paquete;
```

- Una especificación puede existir sin cuerpo pero no al contrario.
- No puede existir un procedimiento en un paquete e individual simultáneamente.

PL/SQL-109

ORACLE®

Especificación del paquete: Ejemplo

```
SQL>CREATE OR REPLACE PACKAGE com_package IS
  2   g_com  NUMBER := 10;      -- inicializado a 10
  3   PROCEDURE reset_com
  4     (v_com      IN      NUMBER);
  5 END com_package;
  6 /
```

```
SQL>EXECUTE com_package.g_com := 5
```

```
SQL>EXECUTE com_package.reset_com(8)
```

PL/SQL-110

ORACLE®

Cuerpo del paquete

Sintaxis

```
CREATE [OR REPLACE] PACKAGE BODY nombre_paquete
IS | AS
    declaraciones de items y tipos privados
    cuerpos de subprogramas
END nombre_paquete;
```

PL/SQL-111

ORACLE®

Cuerpo del paquete: Ejemplo

```
SQL>CREATE OR REPLACE PACKAGE BODY com_package IS
 2 FUNCTION valida_com
 3   (v_com      IN NUMBER) RETURN BOOLEAN
 4 IS
 5   v_max_com NUMBER;
 6 BEGIN
 7   SELECT MAX(com)
 8   INTO   v_max_com
 9   FROM   emp;
10  IF v_com > v_max_com THEN RETURN(FALSE) ;
11  ELSE RETURN(TRUE) ;
12 END IF;
13 END valida_com;
14 END com_package;
15 /
```

PL/SQL-112

ORACLE®

Cuerpo del paquete: Ejemplo

```
SQL>PROCEDURE reset_com
 2 (v_com IN NUMBER)
 3 IS
 4   v_valido BOOLEAN;
 5 BEGIN
 6   v_valido := valida_com(v_com);
 7   IF v_valido = TRUE THEN
 8     g_com := v_com;
 9   ELSE
10     RAISE_APPLICATION_ERROR
11           (-20210, 'Comisión no válida');
12 END IF;
13 END reset_com;
14 END com_package;
15 /
```

PL/SQL-113

ORACLE®

Invocando constructores del paquete

Ejemplo 1

```
SQL> EXECUTE com_package.reset_com(1500);
```

```
SQL> EXECUTE scott.com_package.reset_com(1500);
```

```
SQL> EXECUTE com_package.reset_com@ny (1500);
```

PL/SQL-114

ORACLE®

Invocando constructores del paquete

Ejemplo 2

```
CREATE OR REPLACE PROCEDURE contrato_emp
  (v_nombre IN emp.nombre%TYPE,
   v_mgr    IN emp.mgr%TYPE,
   v_trabajoIN emp.trabajo%TYPE,
   v_sal    IN emp.sal%TYPE)
IS
  v_com  emp.com%TYPE;
. . .
BEGIN
. . .
  v_com := com_package.g_com;
. . .
END contrato_emp;
```

PL/SQL-115

ORACLE®

Borrando paquetes

Para borrar la especificación y cuerpo del paquete:

```
DROP PACKAGE nombre_paquete
```

Para borrar el cuerpo del paquete:

```
DROP PACKAGE BODY nombre_paquete
```

PL/SQL-116

ORACLE®