3º Ingeniero Técnico en Informática de Sistemas

Asignatura: **Fundamentos de Bases de Datos** Curso: **2007/8**

PRÁCTICA T2. El lenguaje procedural *PL/SQL* de Oracle

Objetivos

- Conocer los conceptos y estructuras de programación ofrecidas por PL/SQL, la extensión procedural del SQL de Oracle: estructuras de selección e iteración, cursores, excepciones, bloques anónimos de código, procedimientos, funciones, paquetes (*packages*), disparadores (*triggers*), etc.

Modalidad: laboratorio cerrado (tutorial)

Material de Ayuda

- **Ejemplos** de PL/SQL. (en la página siguiente)
- **Manual de referencia PL/SQL**.
- Manual de referencia de Oracle 10g acerca de **creación de procedimientos, funciones y paquetes**, sobre el manejo de la **salida** para procedimientos y disparadores (*triggers*), y algunos **ejemplos** de programas.

Ejemplos de PL/SQL

- PL/SQL es una extensión procedural de SQL. Está integrado en el núcleo del RDBMS ORACLE.

- Un bloque PL/SQL consta de 3 secciones

```
[DECLARE]
      -- Definición de variables, cursores, ...
BEGIN
      -- Cuerpo
[EXCEPTION]
      -- Excepciones
END
```

- Constantes y variables.

- Tipos de datos
  - NUMBER
  - CHAR (longitud fija)
  - VARCHAR2 (longitud variable)
    - Concatenación ||
  - DATE
    - Funciones sobre fechas
  - BOOLEAN (es un tipo PL/SQL, no de BD).

- Declaración implícita del tipo de datos.
  - Atributo %TYPE
    - var1 var2%TYPE
    - var tabla.campo%TYPE
  - Atributo %ROWTYPE (declaración de registros)
    - var tabla%ROWTYPE
    - var cursor%ROWTYPE
  - Bucles FOR

- Cursores.
  - Implícitos.
    - Manejados (abiertos y cerrados) automáticamente.
    - Nombre predefinido SQL.
    - INSERT, UPDATE, DELETE y SELECT monoregistro.
  - Explícitos.
    - SELECT multiregistro.

- Atributos de los cursores implícitos.
  - SQL%NOTFOUND
  - SQL%FOUND
  - SQL%ROWCOUNT
  - SQL%ISOPEN

- Operaciones con cursores explícitos.
    - DECLARE
        - Cursores con parámetros.
        - Cursores SELECT .. FOR UPDATE
        - UPDATE o DELETE ... WHERE CURRENT OF
    - OPEN
        - Aún no hay filas disponibles.
    - FETCH
        - FETCH cursor INTO lista de variables
        - FETCH cursor INTO registro
    - CLOSE

- Atributos de los cursores explícitos.
    - cursor%NOTFOUND
    - cursor%FOUND
    - cursor%ROWCOUNT
    - cursor%ISOPEN

- Bucles FOR CURSOR.
    FOR var IN cursor [(lista de parámetros] LOOP ... END LOOP;

- Estructuras de programación.
    - Seleccion.
        - IF ... THEN ... ELSIF ... END IF;
        - IS [NOT] NULL
        - IS [NOT] LIKE

    - Iteración (bucles)
        - Bucles LOOP
            LOOP ... END LOOP;
            - Sentencia EXIT
                EXIT [WHEN condición_booleana];

        - Bucles FOR LOOP
            FOR var IN [REVERSE] exp1..exp2 LOOP ... END LOOP;

        - Bucles WHILE LOOP
            WHILE expresión_booleana LOOP ... END LOOP;

        - Bucles FOR cursor.

- Estructura de un bloque PL/SQL
    [DECLARE]
        -- Declaración de constantes, variables,
        -- cursores y excepciones.
    BEGIN
        -- Aquí puede anidarse otro bloque
    [EXCEPTION]
        -- Aquí puede anidarse otro bloque
    END;

- Visibilidad de los identificadores.
    - La normal de LPs con estructura de bloques.

- DML desde bloques PL/SQL.
    - SELECT lista de campos
        INTO lista de variables FROM tablas ...
    - UPDATE ... WHERE CURRENT OF cursor
    - DELETE ... WHERE CURRENT OF cursor

- COMMIT y ROLLBACK.

- Manejo de excepciones.
    - WHEN excepción then tratamiento.

    - Excepciones predefinidas:
        NO_DATA_FOUND, DUP_VAL_ON_INDEX,
        TOO_MANY_ROWS, ZERO_DIVIDE, OTHERS, ...

    - Definición de excepciones.
        DECLARE
            excepción EXCEPTION;
        BEGIN

    - Provocar (elevar) excepciones.
        RAISE excepción
        raise_application_error(-20011, 'Mensaje error')

- Activar la impresión por pantalla
    Escribir en SQL*PLUS, SET SERVEROUTPUT ON

- Impresión por pantalla

```
BEGIN
    DBMS_OUTPUT.DISABLE;
    DBMS_OUTPUT.ENABLE (1000000);
    …
    DBMS_OUTPUT.PUT_LINE (‘texto’);
    DBMS_OUTPUT.PUT (‘texto’);
    DBMS_OUTPUT.NEW_LINE;
END;
```

- Cómo probar el correcto funcionamiento de un procedimiento

```
BEGIN
    Llamada al procedimiento;
END;
```

- Cómo probar el correcto funcionamiento de una función

```
BEGIN
    DBMS_OUTPUT.PUT_LINE (llamada a la función);
END;
```

```
------------------------------------------------------------
DECLARE
  NIF_ VARCHAR2(10);
  NOMBRE_ EMPLEADOS.NOMBRE%TYPE;
  EMP_ EMPLEADOS%ROWTYPE;
  --
  CURSOR C_EMPLEADOS IS
  SELECT NIF, NOMBRE
  FROM EMPLEADOS
  FOR UPDATE OF SUELDO;
  --
  EMP1_ C_EMPLEADOS%ROWTYPE;
BEGIN
  DBMS_OUTPUT.DISABLE;
  DBMS_OUTPUT.ENABLE (1000000);
  --
  OPEN C_EMPLEADOS;
  LOOP
    FETCH C_EMPLEADOS INTO NIF_, NOMBRE_;
    EXIT WHEN C_EMPLEADOS%NOTFOUND;
    --
    DBMS_OUTPUT.PUT_LINE (NOMBRE_);
  END LOOP;
  CLOSE C_EMPLEADOS;
  --
  FOR I IN C_EMPLEADOS LOOP
    UPDATE EMPLEADOS
    SET SUELDO = SUELDO*1.10
    WHERE CURRENT OF C_EMPLEADOS;
    --
    IF (SQL%NOTFOUND) THEN
      RAISE ...
    END IF;
    --
    DBMS_OUTPUT.PUT (I.NOMBRE);
    DBMS_OUTPUT.PUT (I.SUELDO);
    DBMS_OUTPUT.NEW_LINE;
  END LOOP;
END;

------------------------------------------------------------------
CREATE FUNCTION EXISTE_EMPLEADO (
  NIF_ IN VARCHAR2
) RETURN BOOLEAN
IS
  X CHAR(1);
  SORPRESA EXCEPTION;
BEGIN
  SELECT 'X' INTO X FROM EMPLEADOS WHERE NIF = NIF_;
  RETURN (TRUE);
EXCEPTION
  WHEN NO_DATA_FOUND THEN    RETURN (FALSE);
  WHEN TOO_MANY_ROWS THEN    RETURN (TRUE);
  WHEN OTHERS   THEN  RAISE SORPRESA;
END;
```

```
-----------------------------------------------------------------
CREATE TRIGGER COMPROBAR_SUELDO
  BEFORE
  INSERT OR UPDATE OF SALARIO, PUESTO ON EMPLEADOS
  FOR EACH ROW
  WHEN (NEW.PUESTO <> 'PRESIDENTE')
DECLARE
  SUELDO_FUERA_RANGO EXCEPTION;
BEGIN
  IF (:NEW.SALARIO < 100000 OR
      :NEW.SALARIO > 1000000) THEN
    RAISE SUELDO_FUERA_RANGO;
  END IF;
END;

-----------------------------------------------------------------
CREATE PACKAGE emp_actions AS  -- package specification
   PROCEDURE hire_employee (emp_id INTGER, name VARCHAR2, ...);
   PROCEDURE fire_employee (emp_id INTEGER);
   PROCEDURE raise_salary (emp_id INTEGER, increase REAL);
   ...
END emp_actions;

-----------------------------------------------------------------
CREATE PACKAGE BODY emp_actions AS  -- package body
   PROCEDURE hire_employee (emp_id INTGER, name VARCHAR2, ...) IS
   BEGIN
      INSERT INTO emp VALUES (empno, ename, ...);
   END hire_employee;

   PROCEDURE fire_employee (emp_id INTEGER) IS
   BEGIN
      DELETE FROM emp WHERE empno = emp_id;
   END fire_employee;

   PROCEDURE raise_salary (emp_id INTEGER, increase REAL) IS
      salary REAL;
   BEGIN
      SELECT sal INTO salary FROM emp WHERE empno = emp_id;
      ...
   END raise_salary;
   ...
END emp_actions;
```

```
-----------------------------------------------------------------
DECLARE
  CURSOR CDEP IS
  SELECT * FROM DEPARTAMENTOS ORDER BY CODDEP;
  --
  CURSOR CEMP (CODDEP_ DEPARTAMENTOS.CODDEP%TYPE)
  IS
  SELECT * FROM EMPLEADOS
  WHERE CODDEP = CODDEP_
  ORDER BY CODEMP;
BEGIN
  FOR I IN CDEP LOOP
    DBMS_OUTPUT ('DEPARTAMENTO ' || I.CODDEP);
    FOR J IN CEMP (I.CODDEP) LOOP
      DBMS_OUTPUT ('EMPLEADO ' || J.CODEMP);
    END LOOP;
  END LOOP;
END;

-----------------------------------------------------------------
DECLARE
  CURSOR CPROF IS SELECT NIF, NOMBRE FROM PROFESORES ORDER BY NIF;
  CURSOR CALUM IS SELECT NIF, NOMBRE FROM ALUMNOS ORDER BY NIF;
  --
  RPROF CPROF%ROWTYPE;
  RALUM CALUM%ROWTYPE;
BEGIN
  OPEN CPROF;
  OPEN CALUM;
  LOOP
    FETCH CPROF INTO RPROF;
    FETCH CALUM INTO RALUM;
    EXIT WHEN (CPROF%NOTFOUND OR CALUM%NOTFOUND);
    DBMS_OUTPUT.PUT_LINE (RPROF.NIF || ',' || RALUM.NIF);
  END LOOP;
  --
  IF (CPROF%FOUND) THEN
    LOOP
      DBMS_OUTPUT.PUT_LINE (RPROF.NIF || ',');
      FETCH CPROF INTO RPROF;
      EXIT WHEN CPROF%NOTFOUND;
    END LOOP;
  END IF;
  WHILE CALUM%FOUND) LOOP
    DBMS_OTUPUT.PUT_LINE (',' || RALUM.NIF);
    FETCH CALUM INTO RALUM;
  END LOOP;
  CLOSE CPROF;
  CLOSE CALUM;
END;
```

```
-----------------------------------------------------------------
PROCEDURE INSERTAR_O_ACTUALIZAR (
  NIF_    IN EMPLEADOS.NIF%TYPE;
  NOMBRE_ IN EMPLEADOS.NOMBRE%TYPE;
) IS

-- OPCION A -----------------------------------------------------
  X CHAR(1);
BEGIN
  SELECT 'X' INTO X FROM EMPLEADOS WHERE NIF = NIF_;
  UPDATE EMPLEADOS SET NOMBRE = NOMBRE_ WHERE NIF = NIF_;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    INSERT INTO EMPLEADOS (NIF, NOMBRE) VALUES (NIF_, NOMBRE_);
END;

-- OPCION B -----------------------------------------------------
BEGIN
  INSERT INTO EMPLEADOS (NIF, NOMBRE) VALUES (NIF_, NOMBRE_);
EXCEPTION
  WHEN DUP_VAL_ON_INDEX THEN
    UPDATE EMPLEADOS SET NOMBRE = NOMBRE_ WHERE NIF = NIF_;
END;

-- OPCION C -----------------------------------------------------
BEGIN
  UPDATE EMPLEADOS SET NOMBRE = NOMBRE_ WHERE NIF = NIF_;
  IF (SQL%NOTFOUND) THEN
    INSERT INTO EMPLEADOS (NIF, NOMBRE) VALUES (NIF_, NOMBRE_);
  END IF;
END;
```