Ingeniería de Datos

SQL Embebido

Nadia Rodríguez

2012-2

PL/SQL

- Estructura de un Programa PL/SQL
- Declaración de variables
  - Cursores
- Procedimientos
- Funciones
- Triggers
- Paquetes

Estructura de un Programa PL/SQL

DECLARE – *Opcional*

Variables, restricciones, cursores y excepciones definidas por el usuario

BEGIN – *Obligatorio*

Sentencias SQL

Sentencias de control PL/SQL

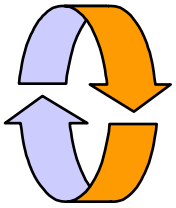
EXCEPTION – *Opcional*

Acciones a ejecutar cuando ocurren errores

END; – *Obligatorio*

Sentencia de Control PL/SQL

- Sentencias condicionales basadas en condiciones Booleanas
  - IF
- Control de Tratamientos Repetitivos Básicos
  - **FOR**
  - **WHILE**



- Manejo de excepciones para manejar errores del DBMS o las excepciones definidas por el usuario.
  - **EXCEPTION**

Estructuras de Control

```
IF condicion1 THEN
        secuencia de instrucciones1;
        ELSIF condition2 THEN
          secuencia de instrucciones2;
        ELSE
          secuencia de instrucciones3;
END IF;

LOOP
        secuencia de instrucciones3;
          ...
END LOOP;

SELECT COUNT(empno)
INTO emp_count
FROM emp;
```

```
WHILE condition
        LOOP
          sequence_of_statements;
          ...
        END LOOP;

FOR contador IN [REVERSE] inicial..final
        LOOP
          secuencia de instrucciones;
          ...
        END LOOP;

LOOP
        EXIT WHEN condicion;
            ...
        END LOOP;
```

Declaración de Variables

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

```
v_estado_civil      CHAR(1);
v_total_sal         NUMBER(9,2) := 0;
v_fecha_orden       DATE := SYSDATE + 7;
v_valido            BOOLEAN NOT NULL := TRUE;
v_cod_cuenta        VARCHAR2(5) NOT NULL := 'AP001';
v_credito_limite    CONSTANT NUMBER := 5000.00;
v_estado            BOOLEAN DEFAULT FALSE;
```

Nuevos Tipos de Datos

```
debito          cuenta.credito%TYPE;
dept_reg        dept%ROWTYPE;

TYPE type_name IS RECORD
    (campo_nombre {tipo_campo | variable%TYPE | tabla.columna%TYPE | tabla%ROWTYPE} [NOT NULL] { := |
    DEFAULT } expr );

    Ejemplo:    TYPE emp_record_type IS RECORD
                (apellido VARCHAR2(25),
                cargo_id VARCHAR2(10),
                salario number(8,2));

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

    Ejemplos:
                TYPE dept_table_type IS TABLE OF
                emp.apellido%TYPE
                INDEX BY BINARY INTEGER;

                TYPE fechecontrato_table_type IS TABLE OF
                DATE
                INDEX BY BINARY INTEGER;
```

Funciones de Manipulación de Caracteres

FUNCION	RESULTADO
CONCAT('Hola', 'Mundo')	HolaMundo
SUBSTR('HolaMundo', 1, 5)	HolaM
LENGTH('HolaMundo')	9
INSTR('HolaMundo', 'M')	5
LPAD(salario, 10, '*')	*****24000
RPAD(salario, 10, '*')	24000*****
TRIM('H' FROM 'HolaMundo')	olaMundo

Funciones Numéricas

FUNCION	RESULTADO
ROUND(45.926, 2)	45.93
TRUNC(45.926, 2)	45.92
MOD(1600, 300)	100

Funciones de Fecha

FUNCION	RESULTADO
MONTHS_BETWEEN(‘01-SEP-95’, ’11-JAN-94’)	19.6774194
ADD_MONTHS(’11-JAN-94’, 6)	11-JUL-94
NEXT_DAY(10-JUN-08’, ‘TUESDAY’)	17-JUN-08
LAST_DAY(’01-FEB-08’)	29-FEB-08
ROUND(SYSDATE, ‘YEAR’ | ‘MONTH’)	?
TRUNC(SYSDATE, ‘YEAR’ | ‘MONTH’)	?

Conversiones de Datos

- Conversión implícita y explícita de datos
- Funciones de Conversión:
  - TO_CHAR
    - TO_CHAR (55000, '$99,999.00')
    - TO_CHAR (trunc(sysdate), 'DD MONTH YYYY')
  - TO_DATE
    - TO_DATE ('25/12/2007', 'DD/MM/YYYY')
    - TO_DATE (01-Jan-90', 'DD-Mon-RR')
  - TO_NUMBER
    - TO_NUMBER('0100')
    - TO_NUMBER('1000.10')

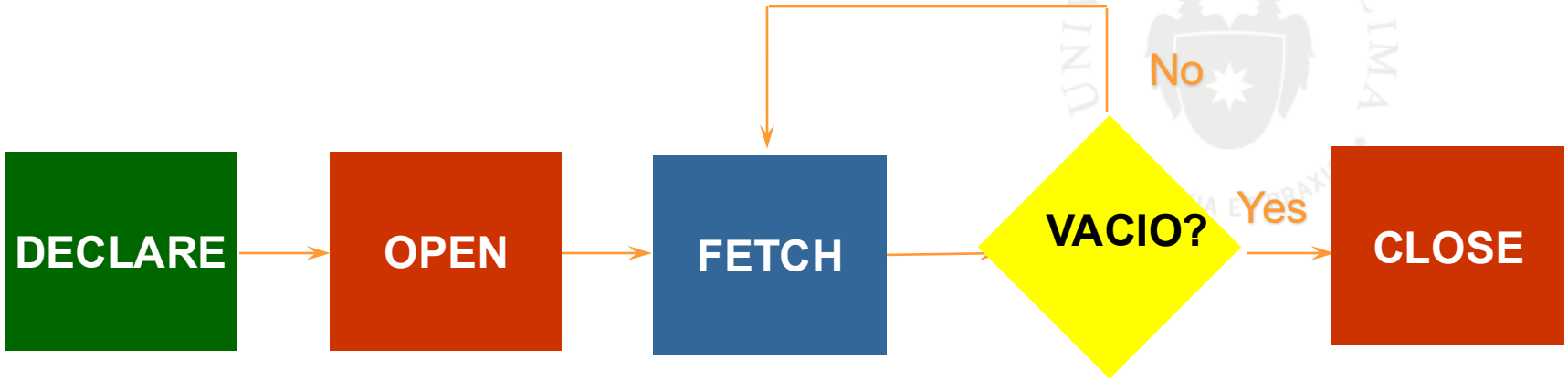
Otras Funciones

- NVL(col|expr1, expr2)
- NVL2(col|expr1, expr2, expr3)
- DECODE(col|expresion,
  busqueda1, resultado1,
  busqueda2, resultado2,
  ……,……,
  default)

Errores Predefinidos

- Excepciones predefinidas del DBMS:
  - NO_DATA_FOUND
  - TOO_MANY_ROWS
  - INVALID_CURSOR
  - ZERO_DIVIDE
  - DUP_VAL_ON_INDEX
  - CURSOR_ALREADY_OPEN
  - VALUE_ERROR
  - PROGRAM_ERROR
  - STORAGE_ERROR

Funcionamiento del Cursor



- Crear el cursor
- Abrir el cursor dentro del programa
- Cargar el valor de la fila en curso del cursor a las variables
- Pregunta si existen más filas
- Regresa al FETCH si hay más filas
- Cerrar el cursor

Cursores Implícitos

Atributo	Tipo	Descripción
%ISOPEN	Booleano	Es TRUE si el cursor está abierto.
%NOTFOUND	Booleano	Es TRUE si el último fetch no ha traido filas.
%FOUND	Booleano	Es TRUE mientras el último fetch trae una fila.
%ROWCOUNT	Numérico	Tiene el número total de filas que han retornado el cursor hasta ese momento.

Programas PL/SQL

- Existen tres categorías principales:
  - Programas simples
  - Procedimientos que ejecutan acciones
  - Funciones que calculan un valor
  - Paquetes que agrupan lógicamente procedimientos y funciones
- Pueden estar almacenados en la Base de datos o desarrollados dentro de una aplicación o subprograma.

Programa Simple utilizando Cursor – Usando FETCH

```
DECLARE
    CURSOR ejecursor IS
        SELECT nombre, salario FROM empleado
        WHERE salario < 5000;
    v_nombre        empleado.nombre%type;
    v_salario       empleado.salario%type;
BEGIN
OPEN ejecursor;    /* apertura el cursor */
LOOP
    FETCH  ejecursor INTO v_nombre, v_salario;  /*asigna los campos de cada linea del cursor a las variables*/
    EXIT WHEN ejecursor%NOTFOUND; /*sale del loop si ya no existen mas líneas en el cursor*/
    INSERT INTO TEMP
      VALUES (v_nombre, v_salario);
    /*inserta las variables obtenidas del cursor en la tabla TEMP*/
END LOOP;
CLOSE ejecursor;    /* cierra cursor */
END;
```

Programa Simple utilizando Cursor – Usando FOR LOOP

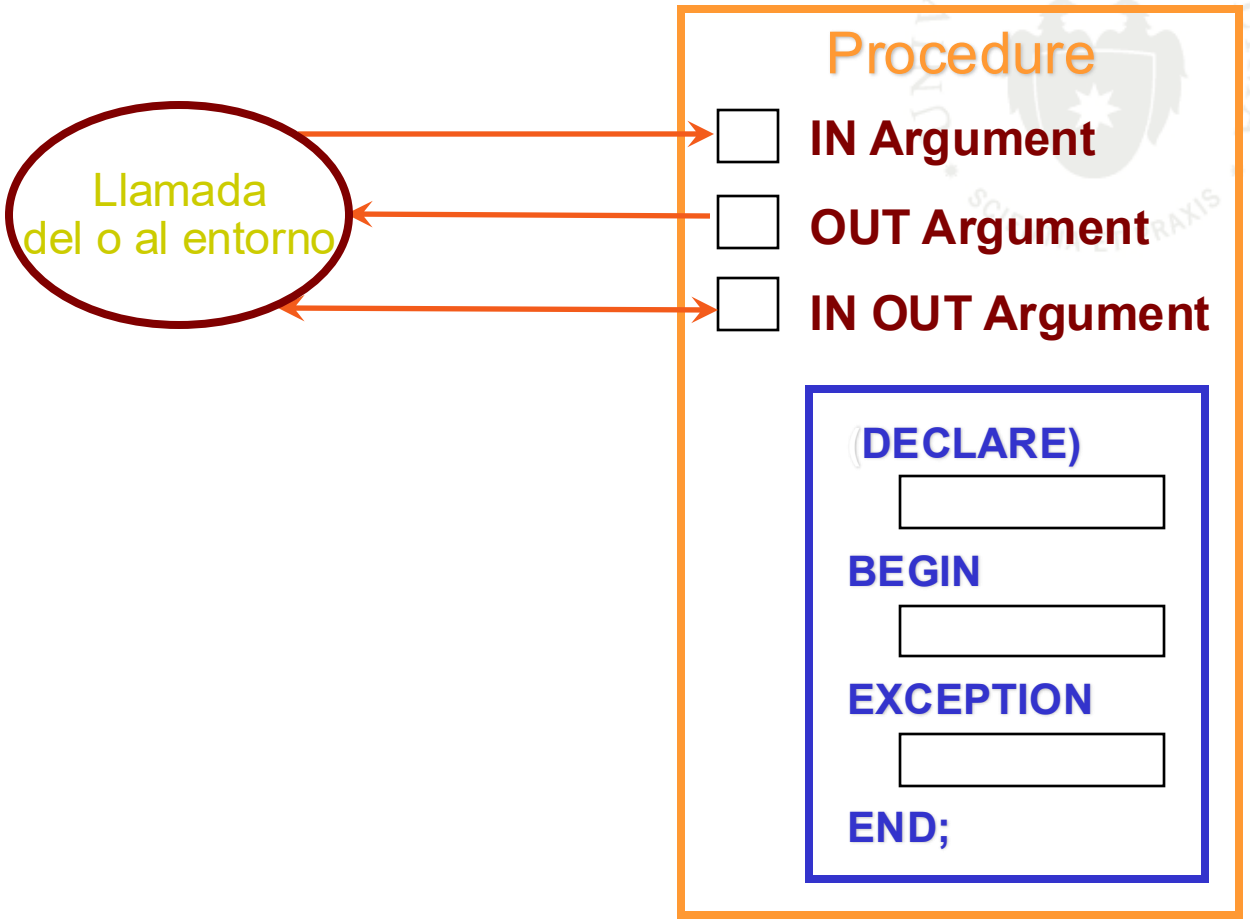
```
DECLARE
    CURSOR ejecursor IS
        SELECT nombre, salario FROM empleado
        WHERE salario < 5000;
BEGIN
FOR ejecursor_rec IN ejecursor
LOOP
    INSERT INTO TEMP
    VALUES (ejecursor_rec.nombre, ejecursor_rec.salario);
  /*inserta las variables obtenidas del cursor en la tabla TEMP*/
END LOOP;
END;
```

Procedimiento

```
CREATE OR REPLACE PROCEDURE <nombre>
     [(parametro,...)]
IS
Bloque_pl/sql;
```

parametro_nombre [IN | OUT | IN OUT] *tipo_dato*[{:= | DEFAULT} *expr*]

Parámetros Procedurales



Creación de un Procedimiento

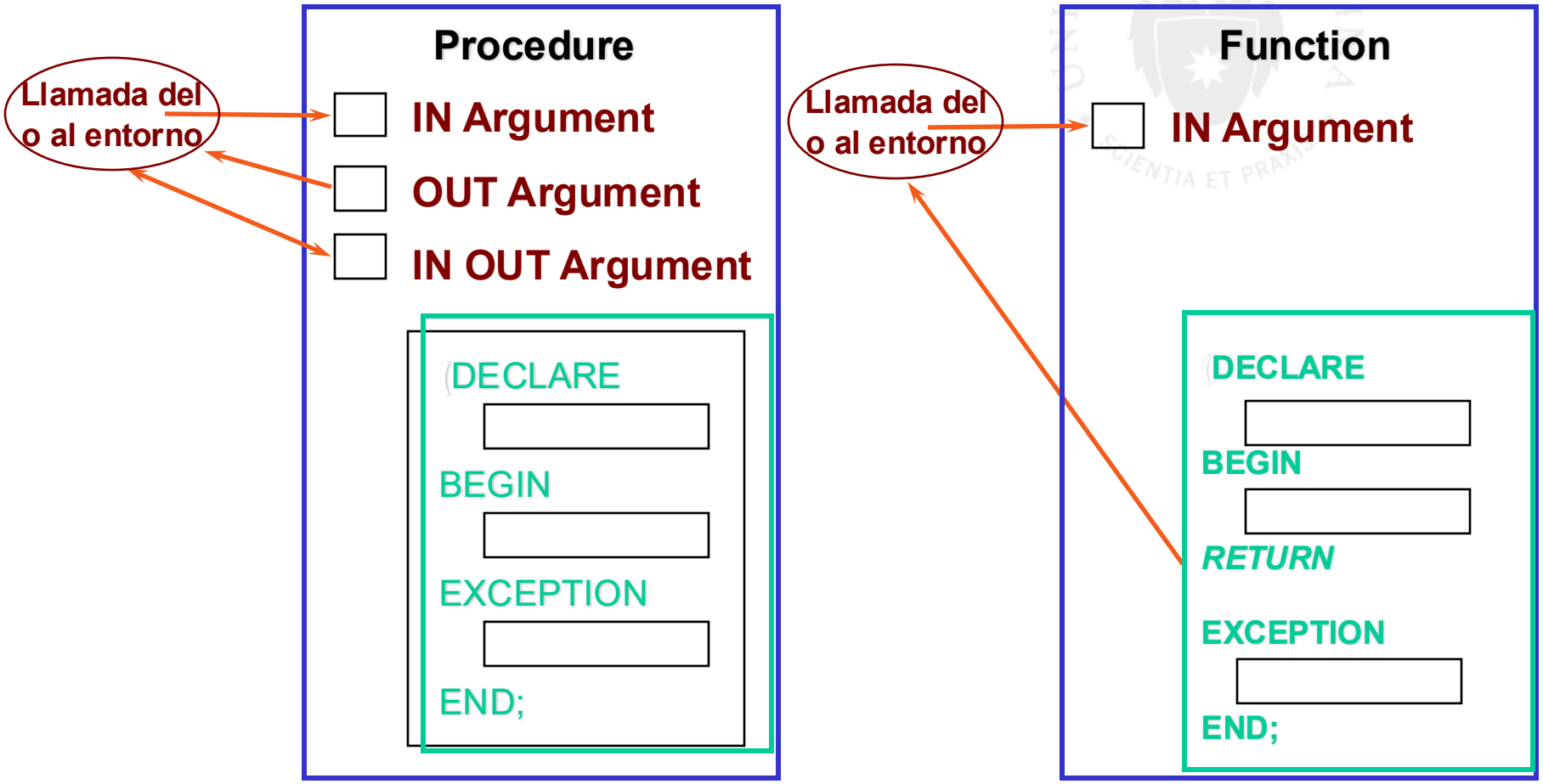
```
CREATE OR REPLACE PROCEDURE cambio_salario
       (v_id_emp IN NUMBER,
        v_nuevo_salario IN NUMBER)
IS
BEGIN
       UPDATE empleado
        SET   salario = v_nuevo_salario
        WHERE cod_empleado = v_id_emp;
       COMMIT;
END cambio_salario;
```

```
SQL> EXECUTE cambio_salario (7839, 12000);
```

Procedimiento

```
CREATE OR REPLACE PROCEDURE actualiza_salario
      (v_id_emp NUMBER, incremento NUMBER)
IS
           salario_actual NUMBER;
           salario_olvidado  EXCEPTION;
BEGIN
          SELECT salary
               INTO salario_actual
              FROM employees
           WHERE employee_id = v_id_emp;
          IF salario_actual IS NULL THEN
            RAISE salario_olvidado;
          ELSE
            UPDATE employees
                  SET salary = salary +  incremento
             WHERE employee_id = v_id_emp;
          END IF;
EXCEPTION
          WHEN NO_DATA_FOUND THEN
              INSERT INTO emp_audit VALUES (v_id_emp, 'Empleado no existe');
          WHEN salario_olvidado THEN
              INSERT INTO emp_audit VALUES (v_id_emp, 'Salario es nulo');
END actualiza_salario;
```

Procedimiento o Función



Función

```
CREATE OR REPLACE FUNCTION <nombre>
      [(parametro,...)]
      RETURN tipo_dato
IS
Bloque_pl/sql;
```

- Retorna un valor al ambiente de donde fue llamado
- Se debe especificar el tipo de dato que debe retornar en la declaración de la función.
- Incluye al final del bloque PL/SQL la cláusula RETURN.

Creación de una Función

```
CREATE OR REPLACE FUNCTION impuesto
      (v_valor IN NUMBER)
      RETURN NUMBER
IS
BEGIN
      RETURN (v_valor * 0.30);
END impuesto;
```

```
PL/SQL> SELECT  salary, impuesto(salary) IMPUESTO
     +> FROM    employees
     +> WHERE   employee_id = XXXX;

      SALARIO  IMPUESTO
------------- ----------
    10000.00      3000.00
```

Función

```
CREATE OR REPLACE FUNCTION salario_ok
    (v_salario NUMBER, v_jobid NUMBER)
     RETURN NUMBER
IS
          min_sal   NUMBER;
          max_sal   NUMBER;
BEGIN
          SELECT min_salary, max_salary
              INTO min_sal, max_sal
            FROM jobs
          WHERE job_id= v_jobid;
IF (v_salario >= min_sal) AND (v_salario <= max_sal) THEN
                RETURN 1;
          ELSE RETURN 0;
          END IF;
END salario_ok;    /*Retorna  1 si es TRUE y 0 si es FALSE según la evaluación  */
```

Trigger

```
CREATE OR REPLACE TRIGGER <nombre>
BEFORE|AFTER INSERT|UPDATE|DELETE OF [columna,.] ON tabla
FOR EACH ROW
WHEN condición
DECLARE
Bloque_pl/sql;
```

VARIABLES

- *:new.columna*
- *:old.columna*

EXCEPCIONES

- *RAISE_APPLICATION_ERROR()*

Trigger

```
CREATE OR REPLACE TRIGGER chequea_salario
BEFORE INSERT OR UPDATE OF salary, job_id ON employees
FOR EACH ROW
WHEN (new.job_id != 'AD_PRES')
DECLARE
        minsal   NUMBER;
        maxsal   NUMBER;
BEGIN
        SELECT min_salary, max_salary
          INTO minsal, maxsal
          FROM jobs
        WHERE job_id = :new.job_id;        /* Obtiene el rango de salarios para un cargo */
        IF (:new.salary < minsal OR :new.salary > maxsal) THEN
           raise_application_error (-20225, 'Salario fuera de rango');
        ELSIF (:new.salary < :old.salary) THEN
           raise_application_error (-20320, 'Incremento negativo');
        ELSIF (:new.salary > 1.1 * :old.salary) THEN
          raise_application_error (-20325, 'Incremento excede el 10%');
    /* Si el salario esta fuera de rango, el incremento es negativo o excede el 10% se invoca una
    excepción */
        END IF;
END chequea_salario;
```

Paquete

```
CREATE PACKAGE emp_acciones  AS   -- Especificación
        PROCEDURE contrata_empleado
          (ename VARCHAR2, job VARCHAR2, mgr NUMBER, catpago NUMBER,                sal
      NUMBER, deptno NUMBER );
        PROCEDURE despide_empleado
          (emp_id NUMBER);
END emp_acciones;
CREATE PACKAGE BODY emp_acciones AS   -- Detalle
        PROCEDURE contrata_empleado
           (ename VARCHAR2, job VARCHAR2, mgr NUMBER, catpago      NUMBER, sal NUMBER,
deptno NUMBER );
        BEGIN
           INSERT INTO empleado
           VALUES (empno_seq.NEXTVAL, ename, job, mgr, SYSDATE, catpago, sal,      deptno );
        END contrata_empleado;

        PROCEDURE despide_empleado (emp_id NUMBER) IS
        BEGIN
           DELETE FROM empleado
           WHERE cod_empleado = emp_id;
        END despide_empleado;
 END emp_acciones;
```

Ingreso de variables en línea

```
SQL> SELECT   nombre
  2     FROM     empleado
  3     WHERE   cod_empleado =
&cod_empleado;

SQL> SELECT   nombre, &&nombre_columna
  2     FROM     empleado
  3     ORDER BY &nombre_columna;
```