

# PL/SQL

Francisco Moreno  
Universidad Nacional

# Subprogramas: Procedimientos

- A **excepción de los triggers**, hasta ahora los bloques PL/SQL que se han presentado son
  - Anónimos (sin nombre)
  - Temporales (no quedan almacenados en la BD)
- Los bloques PL/SQL se pueden almacenar (**garantizar su persistencia**) en la BD mediante subprogramas (**funciones** y **procedimientos**)
- Los subprogramas pueden tener parámetros

# Procedimientos

## Sintaxis:

```
CREATE [OR REPLACE] PROCEDURE  
nombre_procedimiento  
[( par1 [modo] tipo [, par2 [modo]  
  tipo...])]  
IS | AS  
Bloque PL/SQL
```

- La opción REPLACE reemplaza el procedimiento si existe
- Se puede usar AS o IS (son equivalentes)
- El bloque PL/SQL comienza con la palabra BEGIN o con la declaración de las variables locales (**sin usar la palabra DECLARE**).
- Para ver los errores de compilación en **SQL\*Plus** se puede usar el comando **SHOW ERRORS**



Herramientas como SQL Navigator, PL/SQL Developer, entre otras; facilitan mucho las labores de depuración.

- El "modo" especifica el tipo del parámetro así:
  - **IN** (modo predeterminado): Parámetro de entrada.
  - **OUT**: Parámetro de salida. El subprograma devuelve un valor en el parámetro.
  - **IN OUT**: Parámetro de entrada y salida. El subprograma devolverá un valor posiblemente diferente al enviado inicialmente.
- Para declarar los parámetros usar en lo posible %TYPE y %ROWTYPE
- No se puede especificar tamaño para los parámetros en lo que respecta al tipo de datos

## **Ejemplo. Sea la tabla:**

```
DROP TABLE emp;
```

```
CREATE TABLE emp(  
    cod NUMBER(8) PRIMARY KEY,  
    nom VARCHAR2(20),  
    depto NUMBER(3)  
);
```

```
INSERT INTO emp VALUES (12, 'Mel B', 10);
```

```
INSERT INTO emp VALUES (15, 'Mel C', 5);
```

```
INSERT INTO emp VALUES (76, 'Emma Bunton', 15);
```

```
CREATE OR REPLACE PROCEDURE consulta_emp  
(v_nro IN emp.cod%TYPE)
```

```
IS
```

→ Parámetro de entrada

```
v_nom emp.nom%TYPE;
```

```
BEGIN
```

```
    SELECT nom INTO v_nom
```

```
    FROM emp
```

```
    WHERE cod = v_nro;
```

```
    DBMS_OUTPUT.PUT_LINE(v_nom);
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Empleado no existe');
```

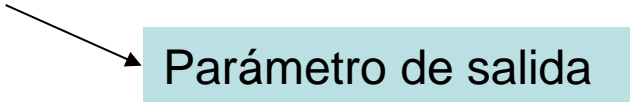
```
END;
```

```
/
```

```
EXECUTE consulta_emp(15);
```

Los parámetros de salida **usualmente son recibidos por variables de otros subprogramas** que los invocan.

```
CREATE OR REPLACE PROCEDURE consulta_emp
(v_nro IN emp.cod%TYPE, v_nom OUT emp.nom%TYPE)
IS
BEGIN
    SELECT nom INTO v_nom -- Se llena el parámetro
    FROM emp
    WHERE cod = v_nro;
    EXCEPTION
    WHEN NO_DATA_FOUND THEN
        v_nom := 'Sin nombre'; -- Si no se llena queda en NULL
    END;
/
```





## Invocación desde otro subprograma:

```
CREATE OR REPLACE PROCEDURE
  invoca_consulta(v_nro IN emp.cod%TYPE)
IS
  nombre emp.nom%TYPE;
BEGIN
  consulta_emp(v_nro, nombre);
  DBMS_OUTPUT.PUT_LINE(' El nombre es: ' || nombre);
END;
/
```

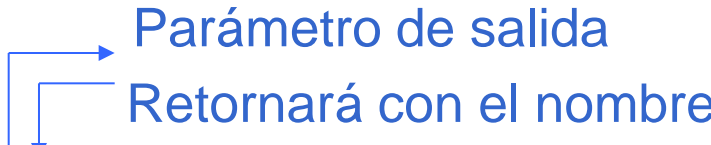


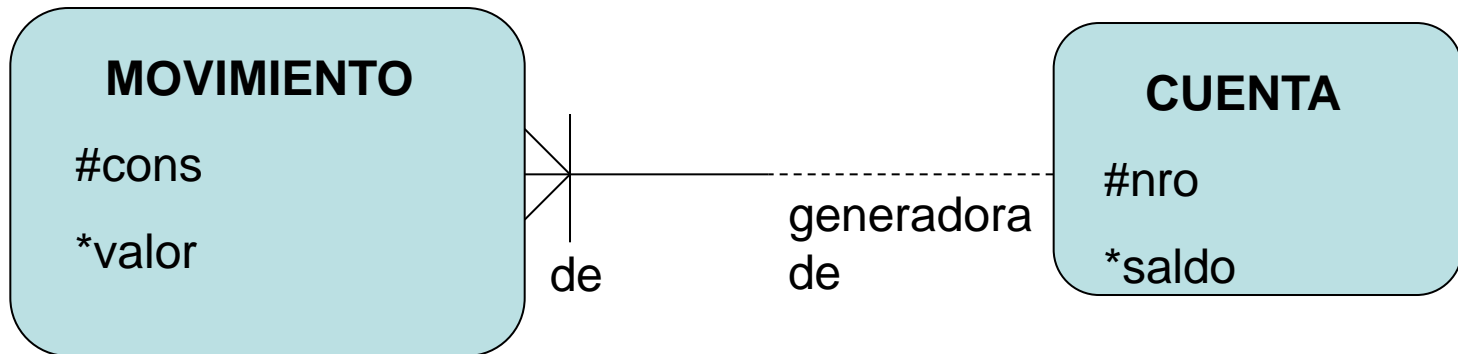
Diagram illustrating the return value of the `consulta_emp` procedure call:

- Parámetro de salida
- Retornará con el nombre

Para ejecutar: `EXECUTE invoca_consulta(15);`

# Ejemplo 3

- Sea el modelo:



## **Sean las tablas:**

```
CREATE TABLE cuenta(  
  nro NUMBER(8) PRIMARY KEY,  
  saldo NUMBER(8) NOT NULL  
);
```

```
INSERT INTO cuenta VALUES (1, 100);  
INSERT INTO cuenta VALUES (2, 400);  
INSERT INTO cuenta VALUES (3, 600);
```

```
CREATE TABLE mvto(  
  cons NUMBER(8),  
  cta NUMBER(8) NOT NULL REFERENCES cuenta,  
  PRIMARY KEY(cons, cta),  
  valor NUMBER(8) NOT NULL  
);
```



```
INSERT INTO mvto VALUES (1, 2, 20);  
INSERT INTO mvto VALUES (2, 2, 20);  
INSERT INTO mvto VALUES (3, 2, 30);  
INSERT INTO mvto VALUES (4, 2, 20);  
INSERT INTO mvto VALUES (5, 2, 20);  
INSERT INTO mvto VALUES (6, 2, 10);
```

```
INSERT INTO mvto VALUES (1, 3, 30);  
INSERT INTO mvto VALUES (2, 3, 10);
```

La cuenta 1 no  
tiene  
movimientos

Desarrollar un procedimiento que imprima **cada cuenta** con sus *n* primeros valores de movimiento (según el cons), en formato horizontal:

Ejemplo: si *n* = 3, la salida debe ser:

|   |          |  |                            |
|---|----------|--|----------------------------|
| 1 |          |    | No tiene movimientos       |
| 2 | 20 20 30 |  |                            |
| 3 | 30 10    |  | Solo tiene dos movimientos |

```
CREATE OR REPLACE PROCEDURE topn_horizontal(n IN POSITIVE) IS
cadena VARCHAR2(1000);
CURSOR ordemov(nro_cta NUMBER) IS Cursor con parámetro
SELECT valor
FROM mvto
WHERE cta = nro_cta AND cons <= n ORDER BY cons ASC;
BEGIN
  FOR mi_c1 IN (SELECT nro FROM cuenta ORDER BY nro) LOOP
    cadena := mi_c1.nro;
    FOR mi_c2 IN ordemov(mi_c1.nro) LOOP
      cadena := cadena || ' ' || mi_c2.valor;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE(cadena);
  END LOOP;
END;
/
```

```
EXECUTE topn_horizontal(3);
```

- ¿La solución usando solo SQL?
- La idea es crear una consulta SQL en la que el usuario **solamente** indica el valor de *n*
- El problema es que en el momento de escribir la consulta no se sabe cuantas columnas se van a necesitar para los *n* primeros valores → Esto se puede solucionar, por ejemplo, con la función de agregación **LISTAGG**. Pero veamos primero una solución para *n* = 2 y *n* = 3.

# SQL: una solución para $n = 2$ y $n = 3$

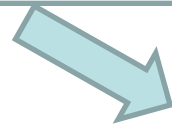
```
SELECT cta || ' ' ||
```

```
MAX(CASE WHEN cons = 1 THEN valor ELSE NULL END) || ' ' ||  
MAX(CASE WHEN cons = 2 THEN valor ELSE NULL END)
```

```
FROM mvto
```

```
WHERE cons <= 2
```

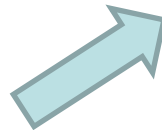
```
GROUP BY cta;
```



Se concatenan **dos** valores

```
SELECT cta || ' ' ||
```

```
MAX(CASE WHEN cons = 1 THEN valor ELSE NULL END) || ' ' ||  
MAX(CASE WHEN cons = 2 THEN valor ELSE NULL END) || ' ' ||  
MAX(CASE WHEN cons = 3 THEN valor ELSE NULL END)
```



Se concatenan **tres** valores

```
FROM mvto
```

```
WHERE cons <= 3
```

```
GROUP BY cta;
```

**Nota:** estas dos soluciones **NO** incluyen las cuentas que no tienen movimientos



# SQL: solución para cualquier **n**

```
SELECT cta || ' ' || LISTAGG(valor, ' ' )  
      WITHIN GROUP (ORDER BY cons ASC)  
      AS listado  
  
FROM mvto  
WHERE cons <= n  
GROUP BY cta  
ORDER BY cta;
```

Cambiar **n** por el  
número deseado

**Nota:** esta solución **NO** incluye las cuentas que no tienen movimientos.  
**Tarea:** Completar la consulta para que incluya esas cuentas.

# Subprogramas: Funciones

- Si un procedimiento tiene solo un parámetro de salida, se puede remplazar por una función y esta se puede involucrar directamente en expresiones y en consultas SQL.
- Sintaxis:

```
CREATE [OR REPLACE] FUNCTION
nombre_función
[(par1 [modo] tipo
[, par2 [modo] tipo...])]
RETURN tipo_de_dato
IS
Bloque de PL/SQL
```

# Ejemplo: Recursividad

```
CREATE OR REPLACE FUNCTION factorial(n IN NUMBER)
RETURN NUMBER IS
BEGIN
  IF n <= 0 THEN --Cuidado con los negativos
    RETURN 1;
  ELSE
    RETURN n*factorial(n-1);
  END IF;
END;
/
```

```
SELECT factorial(5) AS fac
FROM dual;
```