

Cursores explícitos

Consideraciones previas: Únicamente se deben pedir datos por teclado o imprimir mensajes por pantalla en los bloques anónimos, evitando hacerlo dentro de funciones/procedimientos.

1. Crea un procedimiento que reciba un puesto y muestre por pantalla el DNI, el nombre y el salario de todos los mecánicos de ese puesto (utiliza un bucle while). Si no existe mecánicos del puesto indicado se debe mostrar por pantalla que no existen mecánicos de con ese puesto.

```
CREATE OR REPLACE PROCEDURE p_recibePuesto_v1 (V_PUESTO IN MECANICOS.PUESTO%TYPE)
```

```
IS
```

```
    CURSOR c_datos_compuestos IS
```

```
        SELECT *
```

```
        FROM MECANICOS
```

```
        WHERE MECANICOS.PUESTO = V_PUESTO;
```

```
    V_DATOS MECANICOS%ROWTYPE;
```

```
BEGIN
```

```
    OPEN c_datos_compuestos;
```

```
    FETCH c_datos_compuestos INTO V_DATOS;
```

```
    WHILE c_datos_compuestos%FOUND LOOP
```

```
        DBMS_OUTPUT.PUT_LINE(V_DATOS.DNI || ' ' ||
```

```
V_DATOS.NOMBRE || ' ' || V_DATOS.SALARIO);
```

```
        FETCH c_datos_compuestos INTO V_DATOS;
```

```
    END LOOP;
```

```
    CLOSE c_datos_compuestos;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('No existen mecanicos con ese puesto');
```

```
END p_recibePuesto_v1;
```

```
-- bloque anónimo
```

```
DECLARE
```

```
    V_PUESTO1 MECANICOS.PUESTO%TYPE;
```

```
BEGIN
```

```
    V_PUESTO1 := '&PUESTO';
```

```
    p_recibePuesto_v1(V_PUESTO1);
```

```
END;
```

```

-- 1. Crea un procedimiento que reciba un puesto y muestre por pantalla el DNI, el nombre y el sa

CREATE OR REPLACE PROCEDURE p_recibePuesto_v1 (V_PUESTO IN MECANICOS.PUESTO%TYPE)
IS
    CURSOR c_datos_compuestos IS
        SELECT *
        FROM MECANICOS
        WHERE MECANICOS.PUESTO = V_PUESTO;

    V_DATOS MECANICOS%ROWTYPE;

BEGIN
    OPEN c_datos_compuestos;
    FETCH c_datos_compuestos INTO V_DATOS;
    WHILE c_datos_compuestos%FOUND LOOP
        DBMS_OUTPUT.PUT_LINE(V_DATOS.DNI || ' ' || V_DATOS.NOMBRE || ' ' || V_DATOS.SALARIO);
        FETCH c_datos_compuestos INTO V_DATOS;
    END LOOP;
    CLOSE c_datos_compuestos;

EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('No existen mecanicos con ese puesto');

END p_recibePuesto_v1;
  
```

Salida de Script: Tarea terminada en 0,063 segundos

Procedure P_RECIBEPUERTO_V1 compilado

```

DECLARE
    V_PUESTO1 MECANICOS.PUESTO%TYPE;

BEGIN
    V_PUESTO1 := '&PUESTO';

    p_recibePuesto_v1(V_PUESTO1);

END;
  
```

Salida de Script: Tarea terminada en 3,969 segundos

p_recibePuesto_v1(V_PUESTO1);

Salida de DBMS

Tamaño de Buffer: 20000

TALLER12

```

1030 ANER 1110
1001 ANTONIO 600
1002 LUIS 1000
  
```

2. Realiza las acciones que se indican a continuación:

- a. Actualiza el procedimiento anterior para que lance una excepción con el nombre `NO_EXISTEN_MECANICOS` (que hay que declarar previamente) cuando no existan mecánicos con ese puesto. Captura la excepción en el propio procedimiento y pruébalo.
- b. Ahora captura la excepción fuera del procedimiento (en el bloque anónimo desde donde pruebas) ¿Qué ocurre? ¿Por qué?
- c. Utiliza ahora la directiva `RAISE_APPLICATION_ERROR(-20000,'No existen mecánicos con el puesto indicado')` para lanzar la excepción en lugar de `NO_EXISTEN_MECANICOS`. Captura la excepción fuera del procedimiento usando `PRAGMA EXCEPTION_INIT(no_existen_mecanicos,-20000);`

3. Crea un procedimiento que reciba una marca y muestre por pantalla los datos de los coches de esa marca (utiliza un bucle for).

```
CREATE OR REPLACE PROCEDURE p_recibeMarca_v1 (V_MARCA IN COCHES.MARCA%TYPE)
```

IS

```
CURSOR c_datos IS
  SELECT *
  FROM COCHES
  WHERE COCHES.MARCA = V_MARCA;
```

```
V_COCHE COCHES%ROWTYPE;
```

BEGIN

```
FOR V_COCHE IN c_datos LOOP
  DBMS_OUTPUT.PUT_LINE('Matricula: ' || V_COCHE.MATRICULA);
  DBMS_OUTPUT.PUT_LINE('Marca: ' || V_COCHE.MARCA);
  DBMS_OUTPUT.PUT_LINE('Modelo: ' || V_COCHE.MODELO);
  DBMS_OUTPUT.PUT_LINE('Año fabricacion: ' || V_COCHE.AÑO_FABRICACION);
  DBMS_OUTPUT.PUT_LINE(' ');
END LOOP;
```

```
END p_recibeMarca_v1;
```

```
-- bloque anónimo
```

```
DECLARE
  V_MARCA1 COCHES.MARCA%TYPE;
```

```
BEGIN
  V MARCA1 := '&MARCA';
```

```
p_recibeMarca_v1(V_MARCA1);
```

END;

```
-- 3. Crea un procedimiento que reciba una marca y muestre por pantalla los  

CREATE OR REPLACE PROCEDURE p_recibeMarca_v1 (V_MARCA IN COCHES.MARCA%TYPE)  

IS  

    CURSOR c_datos IS  

        SELECT *  

        FROM COCHES  

        WHERE COCHES.MARCA = V_MARCA;  

    V_COCHE COCHES%ROWTYPE;  

BEGIN  

    FOR V_COCHE IN c_datos LOOP  

        DBMS_OUTPUT.PUT_LINE('Matricula: ' || V_COCHE.MATRICULA);  

        DBMS_OUTPUT.PUT_LINE('Marca: ' || V_COCHE.MARCA);  

        DBMS_OUTPUT.PUT_LINE('Modelo: ' || V_COCHE.MODELO);  

        DBMS_OUTPUT.PUT_LINE('Año fabricacion: ' || V_COCHE.AÑO_FABRICACION);  

        DBMS_OUTPUT.PUT_LINE(' ');  

    END LOOP;  

END p_recibeMarca_v1;
```

```

DECLARE
    V_MARCA1 COCHES.MARCA%TYPE;

BEGIN
    V_MARCA1 := '&MARCA1';

    p_recibeMarca_v1(V_MARCA1);

END;

```

Salida de Script x

Tarea terminada en 4,406 segundos

END;

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS

Tamaño de Buffer: 20000

TALLER12 x

Matricula: B4444AC
 Marca: PEUGEOT
 Modelo: 504
 Año fabricacion: 1978

Matricula: CA0000AD
 Marca: PEUGEOT
 Modelo: 205
 Año fabricacion: 1996

Matricula: GR1111AK
 Marca: PEUGEOT
 Modelo: 207
 Año fabricacion: 1998

4. Crear una función denominada HorasPorCoche que reciba la matrícula de un coche y muestre el número de horas que se ha trabajado en ese coche. Trata las excepciones que consideres oportunas.

```
CREATE OR REPLACE FUNCTION f_horasPorCoche (V_MATRICULA IN TRABAJOS.MATRICULA%TYPE)
RETURN NUMBER -- el tipo de dato
```

```
IS
```

```
    CURSOR c_datos IS
        SELECT SUM(HORAS)
        FROM TRABAJOS
        WHERE MATRICULA = V_MATRICULA;
```

```
    V_TOTAL_HORAS NUMBER;
```

```
BEGIN
```

```
    OPEN c_datos;
    FETCH c_datos INTO V_TOTAL_HORAS;
    CLOSE c_datos;
```

```
    RETURN V_TOTAL_HORAS;
```

```
END;
```

```
-- bloque anónimo
```

```
DECLARE
```

```
    V_MATRICULA TRABAJOS.MATRICULA%TYPE;
    V_HORAS_TOTALES NUMBER;
```

```
BEGIN
```

```
    V_MATRICULA := '&MATRICULA';
```

```
    V_HORAS_TOTALES := f_horasPorCoche(V_MATRICULA);
```

```
    DBMS_OUTPUT.PUT_LINE('Horas totales de trabajo invertidas: ' || V_HORAS_TOTALES);
```

```
END;
```

```
-- 4. Crear una función denominada HorasPorCoche que reciba la matrícula de un coche y muestre el número de horas que se ha trabajado en ese coche. Trata las excepciones que consideres oportunas.

CREATE OR REPLACE FUNCTION f_horasPorCoche (V_MATRICULA IN TRABAJOS.MATRICULA%TYPE)
RETURN NUMBER -- el tipo de dato
IS
    CURSOR c_datos IS
        SELECT SUM(HORAS)
        FROM TRABAJOS
        WHERE MATRICULA = V_MATRICULA;

    V_TOTAL_HORAS NUMBER;

BEGIN
    OPEN c_datos;
    FETCH c_datos INTO V_TOTAL_HORAS;
    CLOSE c_datos;

    RETURN V_TOTAL_HORAS;
END;
```

Salida de Script x

Tarea terminada en 0,163 segundos

Function F_HORASPORCOCHE compilado

```
DECLARE
    V_MATRICULA TRABAJOS.MATRICULA%TYPE;
    V_HORAS_TOTALES NUMBER;

BEGIN
    V_MATRICULA := '&MATRICULA';

    V_HORAS_TOTALES := f_horasPorCoche(V_MATRICULA);
    DBMS_OUTPUT.PUT_LINE('Horas totales de trabajo invertidas: ' || V_HORAS_TOTALES);
END;
```

Salida de Script x

Tarea terminada en 1,721 segundos

END;

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS

Tamaño de Buffer: 20000

TALLER12 x

Horas totales de trabajo invertidas: 5,5

5. Crea un procedimiento que utilice la función anterior para mostrar por pantalla las horas trabajadas de todos los coches.

```
CREATE OR REPLACE PROCEDURE p_mostrarHorasTrabajadas
IS
```

```
    CURSOR c_coches IS
        SELECT MATRICULA
        FROM COCHES;
```

```
    V_MATRICULA COCHES.MATRICULA%TYPE;
    V_HORAS NUMBER;
```

```
BEGIN
```

```
    OPEN c_coches;
    FETCH c_coches INTO V_MATRICULA;
```

```
    WHILE c_coches%FOUND LOOP
        V_HORAS :=
        f_horasPorCoche(V_MATRICULA);
```

```
        DBMS_OUTPUT.PUT_LINE('Matricula: ' || V_MATRICULA || ' horas ' || V_HORAS);
        DBMS_OUTPUT.PUT_LINE(' ');
```

```
        FETCH c_coches INTO V_MATRICULA;
    END LOOP;
```

```
END p_mostrarHorasTrabajadas;
```

```
-- bloque anónimo
```

```
DECLARE
```

```
BEGIN
```

```
    p_mostrarHorasTrabajadas;
```

```
END;
```

```
-- 5. Crea un procedimiento que utilice la función anterior para mostrar por pantalla las horas trabajadas de todos los coches.
CREATE OR REPLACE PROCEDURE p_mostrarHorasTrabajadas
IS
    CURSOR c_coches IS
        SELECT MATRICULA --usamos la matricula como truco de a que es una FK
        FROM COCHES;

    V_MATRICULA COCHES.MATRICULA%TYPE;
    V_HORAS NUMBER;

BEGIN
    OPEN c_coches;
    FETCH c_coches INTO V_MATRICULA; -- que vaya recorriendo

    WHILE c_coches%FOUND LOOP
        V_HORAS := f_horasPorCoche(V_MATRICULA); -- porque ahora la matricula a la funcion se la da el cursor c_coches
        DBMS_OUTPUT.PUT_LINE('Matricula: ' || V_MATRICULA || ' horas ' || V_HORAS);
        DBMS_OUTPUT.PUT_LINE(' ');

        FETCH c_coches INTO V_MATRICULA;
    END LOOP;

END p_mostrarHorasTrabajadas;
```

Salida de Script x

Tarea terminada en 0,134 segundos

Procedure P_MOSTRAHORASTRABAJADAS compilado

```
DECLARE

BEGIN
    p_mostrarHorasTrabajadas;

END;
```

Salida de Script x

Tarea terminada en 0,064 segundos

Procedimiento PL/SQL terminado correctamente.

Salida de DBMS

Tamaño de Buffer: 20000

TALLER12 x

```
Matricula: B4444AC horas 3,2

Matricula: CA0000AD horas 8

Matricula: GR1111AK horas 9

Matricula: GR4321A horas 2,1

Matricula: J1234Z horas 12,2

Matricula: J9999AB horas 9,2

Matricula: M3020KY horas 5,5

Matricula: Z199AB horas
```

6. Crea un procedimiento que muestre los puestos de los mecánicos y el número de mecánicos de cada puesto.

```
CREATE OR REPLACE PROCEDURE p_mostrarMecanicos
IS
```

```
  V_PUESTO MECANICOS.PUESTO%TYPE;
  V_NUM_MECANICOS NUMBER;
```

```
  CURSOR c_puesto IS
    SELECT DISTINCT PUESTO
    FROM MECANICOS
    WHERE PUESTO IS NOT NULL;
```

```
  CURSOR c_num_mecanicos IS
    SELECT COUNT(*)
    FROM MECANICOS
    WHERE PUESTO = V_PUESTO;
```

```
BEGIN
```

```
  OPEN c_puesto;
  FETCH c_puesto INTO V_PUESTO;
```

```
  WHILE c_puesto%FOUND LOOP
    OPEN c_num_mecanicos;
    FETCH c_num_mecanicos INTO V_NUM_MECANICOS;
    DBMS_OUTPUT.PUT_LINE
('Puesto: ' || V_PUESTO || ' Nº de Mecanicos: ' || V_NUM_MECANICOS);
    DBMS_OUTPUT.PUT_LINE(' ');
```

```
    FETCH c_puesto INTO V_PUESTO;
    CLOSE c_num_mecanicos;
  END LOOP;
```

```
  CLOSE c_puesto;
```

```
END p_mostrarMecanicos;
```

```
-- bloque anónimo
```

```
DECLARE
```

```
BEGIN
```

```
  p_mostrarMecanicos;
```

```
END;
```

```
CREATE OR REPLACE PROCEDURE p_mostrarMecanicos
IS
  V_PUESTO MECANICOS.PUESTO%TYPE;
  V_NUM_MECANICOS NUMBER;

  CURSOR c_puesto IS
    SELECT DISTINCT PUESTO -- como no es PK se repiten algunos
    FROM MECANICOS
    WHERE PUESTO IS NOT NULL; -- porque hay dos que son nulls...

  CURSOR c_num_mecanicos IS -- parecido a la funcion anterior de la matricula
    SELECT COUNT(*)
    FROM MECANICOS
    WHERE PUESTO = V_PUESTO; -- para que solo cuente los que hay por cada puesto

BEGIN
  OPEN c_puesto;
  FETCH c_puesto INTO V_PUESTO;

  WHILE c_puesto%FOUND LOOP

    OPEN c_num_mecanicos;
    FETCH c_num_mecanicos INTO V_NUM_MECANICOS;
    DBMS_OUTPUT.PUT_LINE('Puesto: ' || V_PUESTO || ' Nº de Mecanicos: ' || V_NUM_MECANICOS);
    DBMS_OUTPUT.PUT_LINE(' ');

    FETCH c_puesto INTO V_PUESTO; -- esto es para que el while siga para alente
    -- aqui es como si faltase el fetch final de c_num_mecanicos
    CLOSE c_num_mecanicos; -- para resetear el count que va haciendo este cursor

  END LOOP;

  CLOSE c_puesto;

END p_mostrarMecanicos;
```

Salida de Script x
Tarea terminada en 0,188 segundos
Procedure P_MOSTRARMECANICOS compilado

```
DECLARE

BEGIN
  p_mostrarMecanicos;

END;
```

Salida de Script x
Tarea terminada en 0,085 segundos
Procedimiento PL/SQL terminado correctamente.

Salida de DBMS
Tamaño de Buffer: 20000

TALLER12 x
Puesto: MOTOR Nº de Mecanicos: 3
Puesto: AMORTIGUACION Nº de Mecanicos: 2
Puesto: CHAPA Nº de Mecanicos: 6