

PRÁCTICA #3

PL/SQL



► Ejercicios-Sesión#1



EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

CURSOR-WHILE

Seleccionamos y ejecutamos el cursor

1 SET SERVEROUTPUT ON;
2 DECLARE
3 CURSOR cursor_alumno IS
4 SELECT * FROM EI.ALUMNO WHERE lugar <> 'Huelva';
5 reg_alumno EI.ALUMNO%ROWTYPE;
6 BEGIN
7 OPEN cursor_alumno;
8 FETCH cursor_alumno INTO reg_alumno;
9 WHILE cursor_alumno%FOUND and cursor_alumno%ROWCOUNT <= 20 LOOP
10 dbms_output.put_line('Nombre: ' || rpad(reg_alumno.nombre,25) || ' Localidad: ' || rpad(reg_alumno.lugar,10));
11 FETCH cursor_alumno INTO reg_alumno;
12 END LOOP;
13 CLOSE cursor_alumno;
14 END;

Escribimos el script PL/SQL

Salida de Script x

Tarea terminada en 0,144 segundos

Nombre: Francisco Gallego Macías	Localidad: Cádiz
Nombre: Teresa Díaz Camacho	Localidad: Madrid
Nombre: Beatriz Rico Vázquez	Localidad: Cádiz
Nombre: Antonio Resines Pérez	Localidad: Córdoba
Nombre: Eva García Gil	Localidad: Cádiz
Nombre: Pablo Gómez Ruiz	Localidad: Córdoba

Procedimiento SQL terminado correctamente.






Verificamos que la ejecución finalizó correctamente



EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

2

Seleccionamos y ejecutamos el cursor

CURSOR-FOR

```
1 SET SERVEROUTPUT ON;
2 DECLARE
3     CURSOR c_alumno IS
4         SELECT * FROM EI.ALUMNO WHERE lugar <> 'Huelva';
5 BEGIN
6     FOR tupla IN c_alumno LOOP
7         dbms_output.put_line('Nombre: ' || rpad(tupla.nombre,25) || ' Localidad: ' || rpad(tupla.lugar,10));
8     END LOOP;
9 END;
```

Salida de Script x

Tarea terminada en 0,173 segundos

Nombre: Teresa Díaz Camacho	Localidad: Madrid
Nombre: Beatriz Rico Vázquez	Localidad: Cádiz
Nombre: Antonio Resines Pérez	Localidad: Córdoba
Nombre: Eva García Gil	Localidad: Cádiz
Nombre: Pablo Gómez Ruíz	Localidad: Córdoba





Procedimiento PL/SQL terminado correctamente.



EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

 FUNCIÓN

Seleccionamos
y ejecutamos
la función



```
NombreAlumno.sql x Sin Título3.sql x
SET SERVEROUTPUT ON;
create or replace FUNCTION NombreAlumno(idAlumno EI.ALUMNO.nAl%TYPE)
RETURN EI.ALUMNO.nombre%TYPE IS
3 nombre_buscado EI.ALUMNO.nombre%TYPE;
4 alumno_muy_antiguo EXCEPTION;
5 BEGIN
6 IF (TO_NUMBER(idAlumno)<100) THEN
7     RAISE alumno_muy_antiguo;
8 END IF;
9 SELECT nombre INTO nombre_buscado FROM EI.ALUMNO ALUM WHERE ALUM.nAl=idAlumno;
10 RETURN nombre_buscado;
11
12 EXCEPTION
13 WHEN alumno_muy_antiguo THEN
14     dbms_output.put_line('Alumno muy antiguo!! :0');
15 WHEN OTHERS THEN
16     dbms_output.put_line('Se produjo un error al ejecutar la función!! :(');
17
18 END NombreAlumno;
```

Salida de Script x

Tarea terminada en 0,19 se

Function NOMBREALUMNO compilado

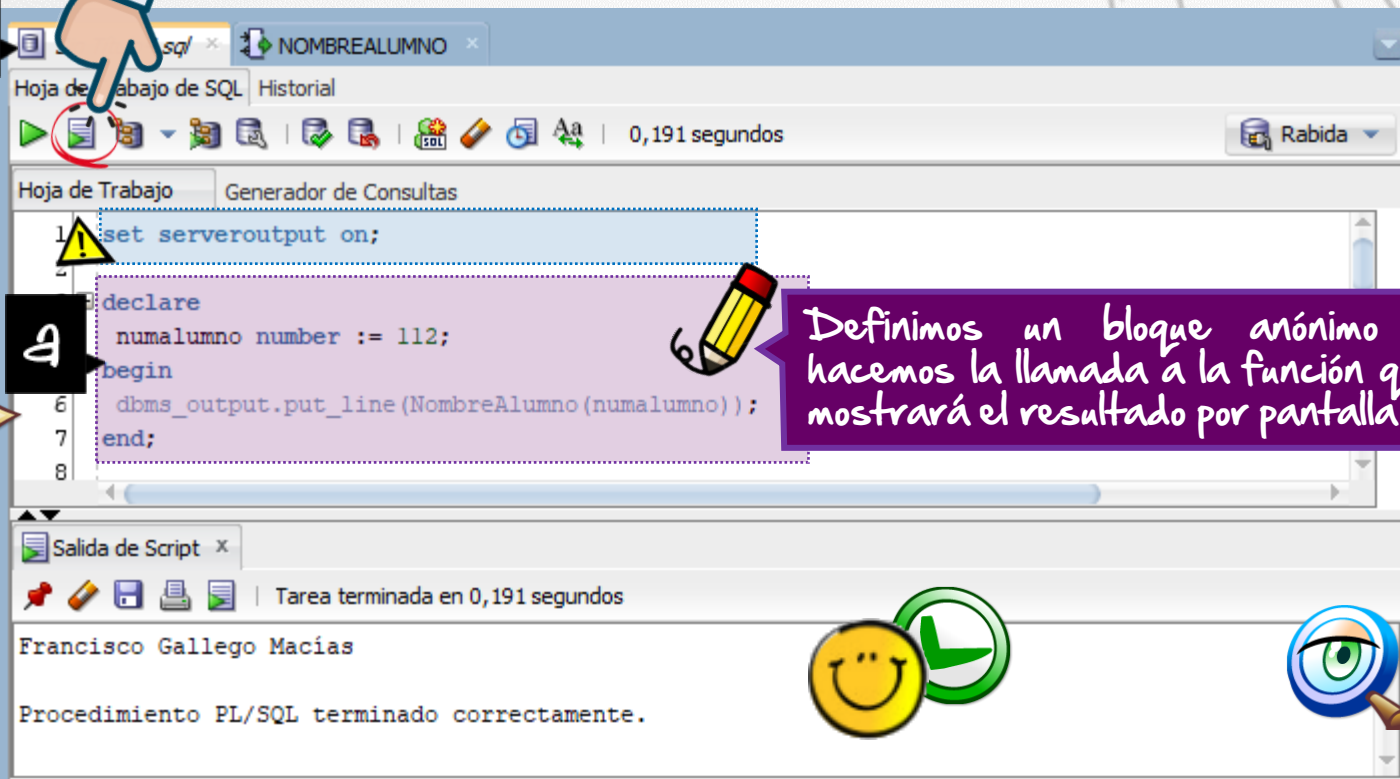
Al ejecutar la función, verificamos
que la compilación de la función
finalizó correctamente, con lo que
se agregará a nuestra colección
de funciones personal



EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

Ejecutamos la función anónima seleccionada

► **FUNCIÓN** Bloque anónimo:



The screenshot shows the SQL Developer interface with a script titled 'NOMBREALUMNO'. The script contains the following code:

```
1 set serveroutput on;  
2  
3 declare  
4   numalumno number := 112;  
5 begin  
6   dbms_output.put_line(NombreAlumno(numalumno));  
7 end;  
8
```

The 'Salida de Script' (Script Output) window at the bottom shows the execution result:

```
Tarea terminada en 0,191 segundos  
Francisco Gallego Macías  
Procedimiento PL/SQL terminado correctamente.
```

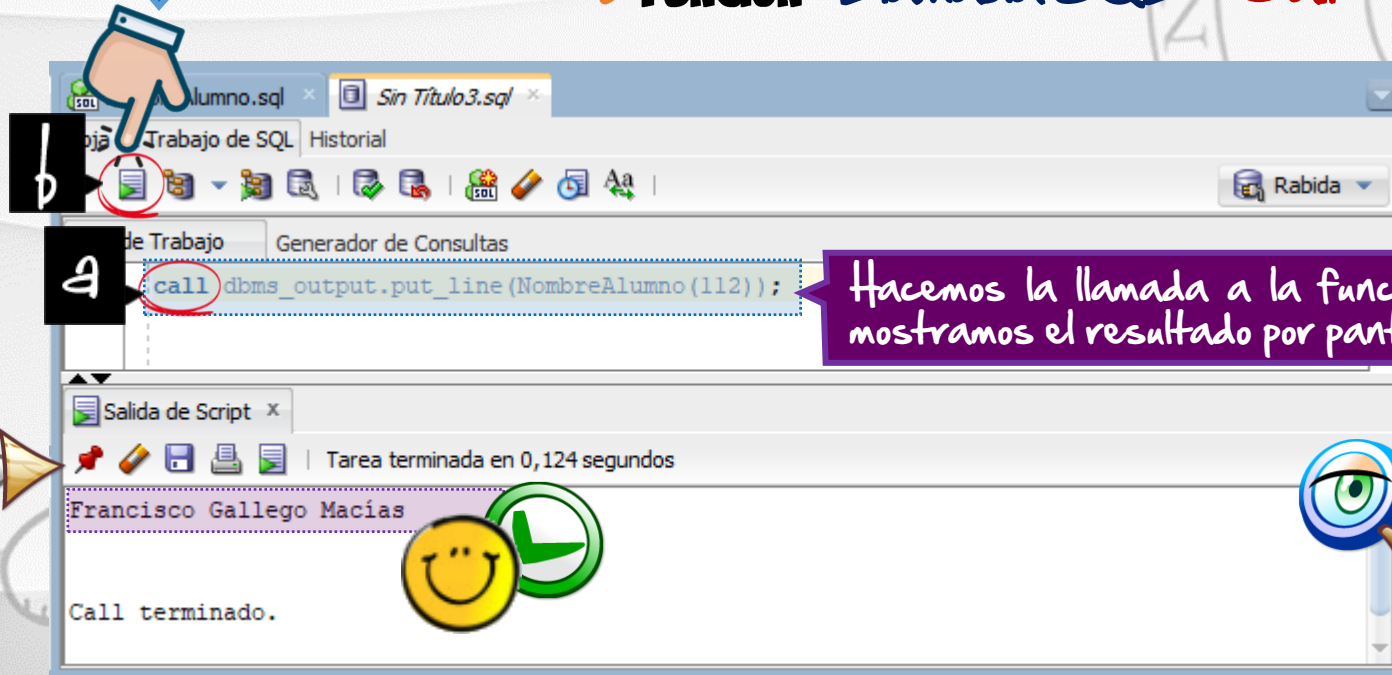
Annotations on the image include:

- A hand icon pointing to the 'Run' button (a green play icon) in the toolbar.
- A pencil icon pointing to the PL/SQL code block.
- A purple callout box stating: 'Definimos un bloque anónimo y hacemos la llamada a la función que mostrará el resultado por pantalla'.
- A yellow smiley face and a green clock icon in the bottom right corner.
- A magnifying glass icon in the bottom right corner.

➔ EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

Ejecutamos la función

▶ **FUNCIÓN** Llamada SQL - Call



Trabajo de SQL Historial

Generador de Consultas

```
call dbms_output.put_line (NombreAlumno (112)) ;
```

Salida de Script x

Tarea terminada en 0,124 segundos

Francisco Gallego Macías

Call terminado.



EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

Seleccionamos y ejecutamos el procedimiento

PROCEDIMIENTO

1

```
SET SERVEROUTPUT ON;  
2 CREATE or REPLACE PROCEDURE ImprimeAlumno(idAlumno EI.ALUMNO.nAl%TYPE) IS  
3   reg_alumno EI.ALUMNO%ROWTYPE;  
4 BEGIN  
5   SELECT * INTO reg_alumno FROM EI.ALUMNO ALUM WHERE ALUM.nAl=idAlumno;  
6   dbms_output.put_line('NAL' || ' ' DNI ' ' || ' ' Nombre');  
7   dbms_output.put_line(rpad(reg_alumno.NAL,3) || ' ' || reg_alumno.DNI || ' ' || rpad(reg_alumno.Nombre,25));  
8 END ImprimeAlumno;
```

Escribimos el procedimiento PL/SQL

3

Salida de Script x

Tarea terminada en 0,185

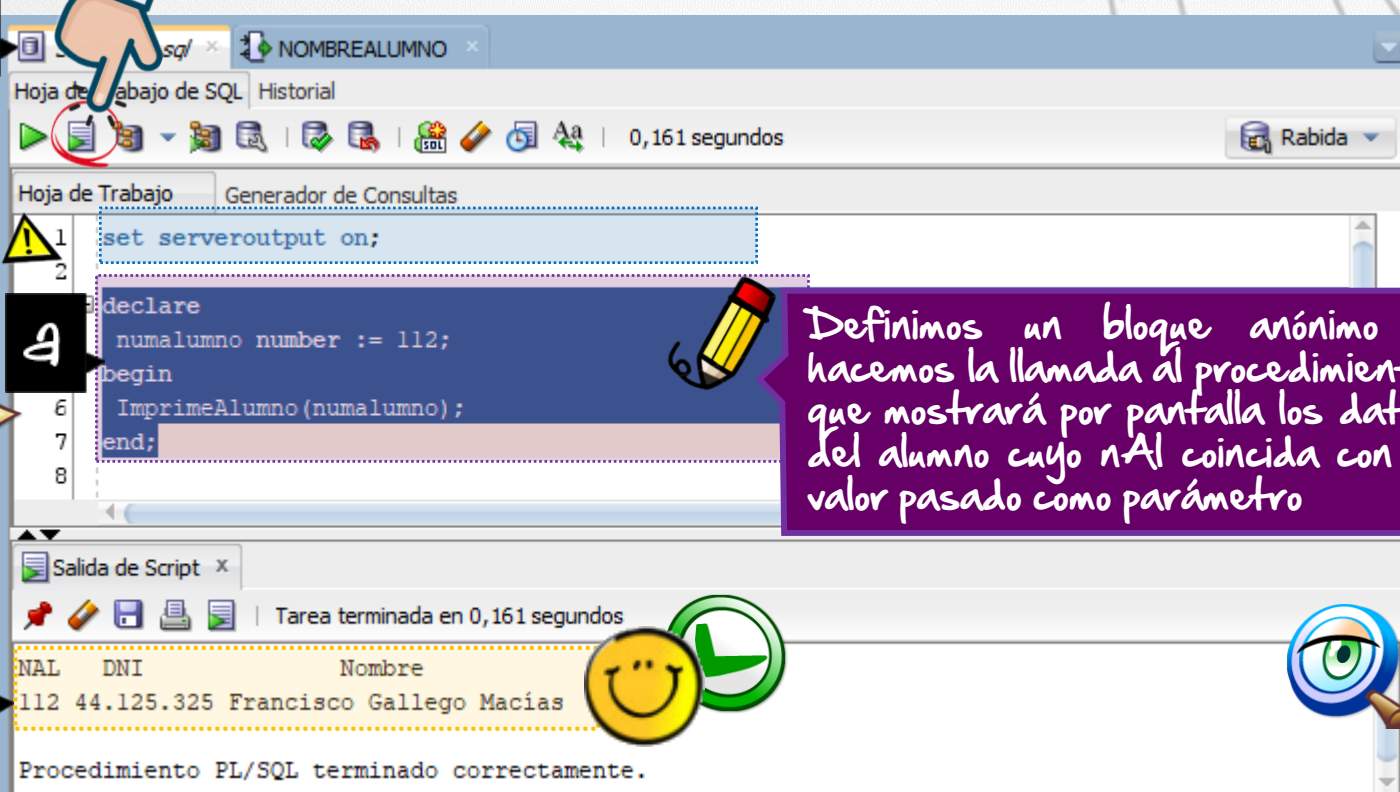
Procedure IMPRIMEALUMNO compilado

Al ejecutar el procedimiento, verificamos que su compilación finalizó correctamente, con lo que se agregará a nuestra colección de procedimientos personal

EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

Ejecutamos la función anónima seleccionada

▶ **PROCEDIMIENTO** Bloque anónimo:



The screenshot shows the SQL Developer interface with the following components:

- Hoja de Trabajo de SQL:** Contains the SQL script being executed.
- Script Content:**

```
1 set serveroutput on;  
2  
3 declare  
4     numalumno number := 112;  
5  
6     ImprimeAlumno(numalumno);  
7 end;  
8
```
- Salida de Script:** Shows the output of the script execution.
- Output Table:**

NAL	DNI	Nombre
112	44.125.325	Francisco Gallego Macías
- Status:** Procedimiento PL/SQL terminado correctamente.

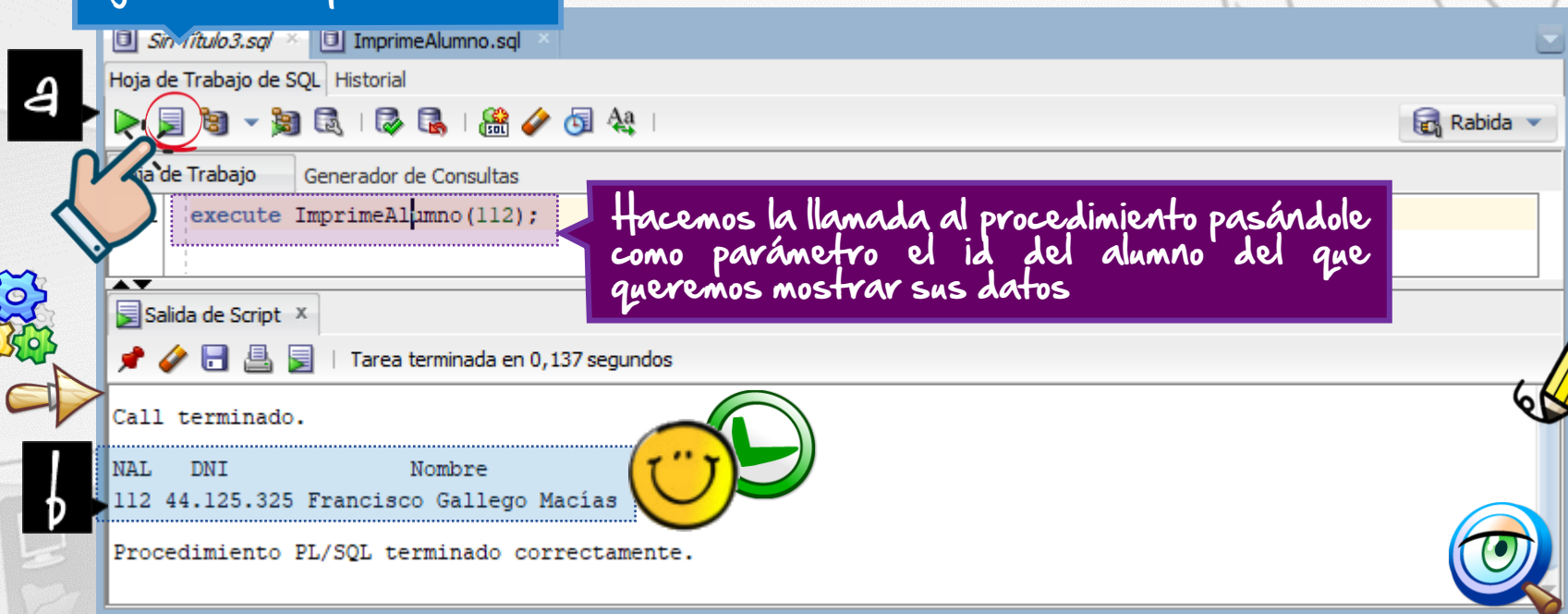
Annotations and icons:

- Hand icon:** Points to the 'Run' button in the SQL Developer toolbar.
- Gears icon:** Located on the left side of the interface.
- Pencil icon:** Points to the SQL script content.
- Smiley face and checkmark icon:** Located next to the output table.
- Magnifying glass icon:** Located at the bottom right of the output area.

➡ EJECUTAR SCRIPTS DESDE SQL DEVELOPER:

▶ **PROCEDIMIENTO** Llamada SQL - *Execute*

Ejecutamos el procedimiento



The screenshot shows the SQL Developer interface with the following elements:

- Hoja de Trabajo de SQL:** Contains the SQL script `execute ImprimeAlumno(112);`. A hand icon points to the `execute` keyword.
- Salida de Script:** Displays the output of the execution, including a table of student data and a confirmation message.
- Annotations:** A purple callout box explains the purpose of the script, and a green checkmark icon indicates successful execution.

Hacemos la llamada al procedimiento pasándole como parámetro el id del alumno del que queremos mostrar sus datos

Call terminado.

NAL	DNI	Nombre
112	44.125.325	Francisco Gallego Macías

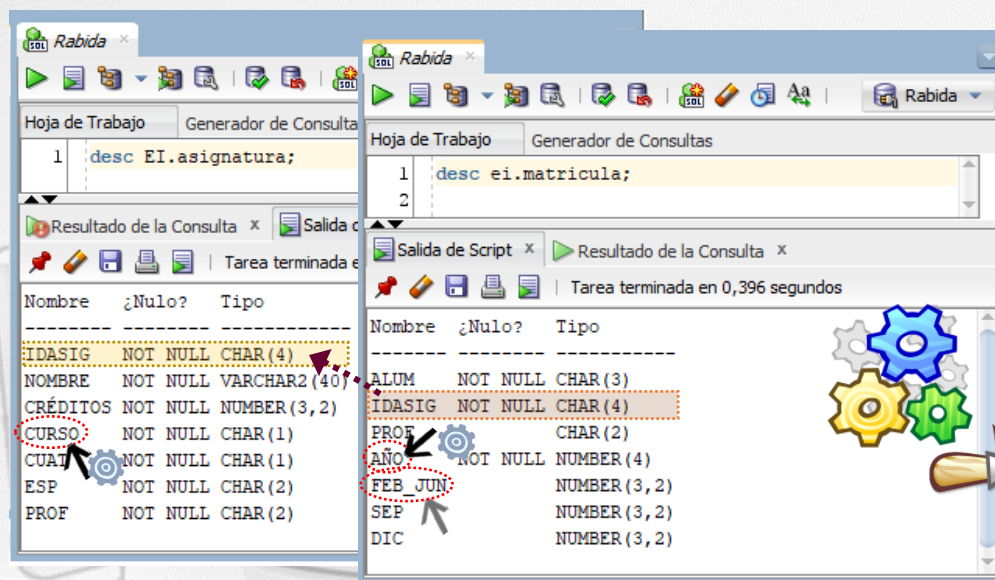
Procedimiento PL/SQL terminado correctamente.

1. PL/SQL – EJEMPLO RESUMEN

➡ Mediante un procedimiento almacenado, se quiere mostrar una estadística por cada asignatura, del porcentaje de alumnos aprobados respecto a los presentados en la convocatoria feb_jun, para un curso y un año académico dados como parámetros.

▶ Por ejemplo, la llamada **porcentaje_aprobados(2,2002)**, mostraría los porcentajes de las asignaturas del **2º curso** en el año 2002.

➡ TABLAS QUE INTERVIENEN EN LA CONSULTA:

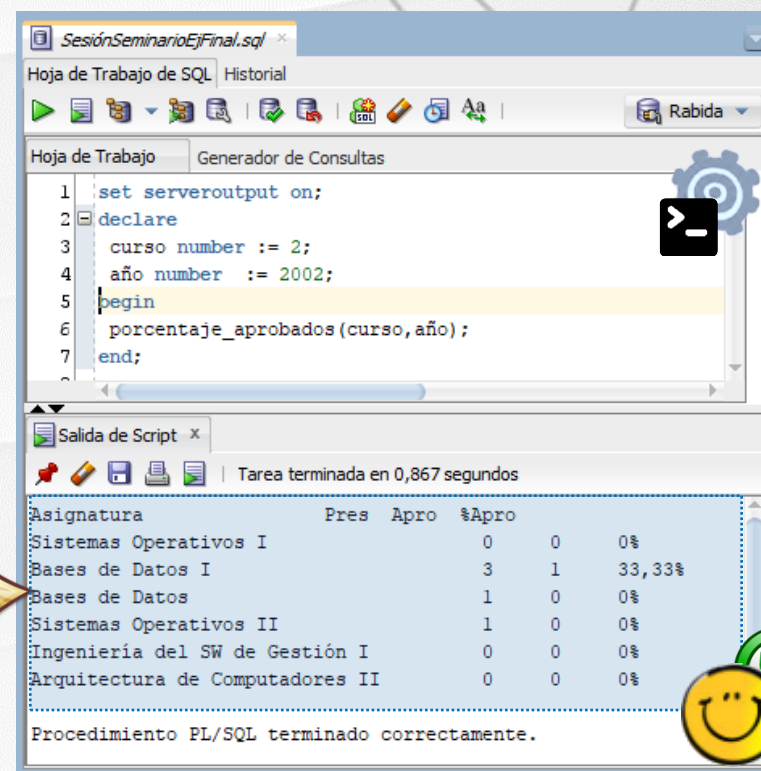


Two screenshots of SQL Developer showing table structures:

- Left Screenshot (EI.asignatura):**

Nombre	¿Nulo?	Tipo
IDASIG	NOT NULL	CHAR(4)
NOMBRE	NOT NULL	VARCHAR2(40)
CRÉDITOS	NOT NULL	NUMBER(3,2)
CURSO	NOT NULL	CHAR(1)
CUAT	NOT NULL	CHAR(1)
ESP	NOT NULL	CHAR(2)
PROF	NOT NULL	CHAR(2)
- Right Screenshot (ei.matricula):**

Nombre	¿Nulo?	Tipo
ALUM	NOT NULL	CHAR(3)
IDASIG	NOT NULL	CHAR(4)
PROF		CHAR(2)
AÑO	NOT NULL	NUMBER(4)
FEB JUN		NUMBER(3,2)
SEP		NUMBER(3,2)
DIC		NUMBER(3,2)



SQL Developer window showing a PL/SQL script and its execution results:

```

1 set serveroutput on;
2 declare
3   curso number := 2;
4   año number := 2002;
5 begin
6   porcentaje_aprobados(curso,año);
7 end;
    
```

Execution results (Salida de Script):

Asignatura	Pres	Apro	%Apro
Sistemas Operativos I	0	0	0%
Bases de Datos I	3	1	33,33%
Bases de Datos	1	0	0%
Sistemas Operativos II	1	0	0%
Ingeniería del SW de Gestión I	0	0	0%
Arquitectura de Computadores II	0	0	0%

Procedimiento PL/SQL terminado correctamente.



→ CURSOR C_ASIGNATURA

1

SesiónSeminarioEjFinal.sql

Hoja de Trabajo de SQL Historial

Hoja de Trabajo Generador de Consultas

```
22 select idAsig, nombre
23 from EI.ASIGNATURA
24 where curso = 2;
```

Salida de Script

Tarea terminada en 0,311 seg.

A003 Sistemas Operativos I

A004 Bases de Datos I

A006 Bases de Datos

A009 Sistemas Operativos II

A010 Ingeniería del SW de Gestión I

A013 Arquitectura de Computadores II

6 filas seleccionadas.

idAsig

P_curso (PROC.)

Lista de asignaturas del 2º curso

#Presentados= 3
#Aprobados= 1
%Aprobados= 33.33%

1

SesiónSeminarioEjFinal.sql

Hoja de Trabajo de SQL Historial

Hoja de Trabajo Generador de Consultas

```
26 cursor c_matriculas(p_idAsig EI.MATRICULA.idAsig%type) is
27 select idAsig, feb_jun
28 from EI.MATRICULA
29 where año=2002 and idAsig='A003';
```

Salida de Script

Tarea terminada en 0,16 segundos

no se ha seleccionado ninguna fila

→ CURSOR C_MATRICULA('A003')

#Presentados= 0
#Aprobados= 0
%Aprobados= 0

2

SesiónSeminarioEjFinal.sql

Hoja de Trabajo de SQL Historial

Hoja de Trabajo Generador de Consultas

```
26 cursor c_matriculas(p_idAsig EI.MATRICULA.idAsig%type) is
27 select idAsig, feb_jun
28 from EI.MATRICULA
29 where año=2002 and idAsig='A004';
```

Salida de Script

Tarea terminada en 0,159 segundos

IDAS	FEB_JUN
A004	3,5
A004	3
A004	7

→ CURSOR C_MATRICULA('A004')

Notas de los alumnos en la asignatura BDI (A004), en la convocatoria feb_jun de 2002

FOR2

FOR1



→ **CURS**OR **C_MATRÍCULA**('A004',2002)

1

2

SesiónSeminarioEjFinal.sql

Hoja de Trabajo de SQL Historial

0,15899999 segundos

Hoja de Trabajo Generador de Consultas

```

25
26 cursor c_matriculas(p_idAsig EI.MATRÍCULA.idAsig&type) is
27 select idAsig, feb_jun
28 from EI.MATRÍCULA
29 where año=2002 and idAsig='A004';
30

```

Salida de Script

Tarea terminada en 0,159 segundos

IDAS	FEB_JUN
A004	NULL
A004	3,5
A004	3
A004	7

CURSOR **C_ASIGNATURA**

FOR1

Para cada tupla del Cursor **c_asignaturas** hacer
#aprobados = 0
#presentados = 0

CORRELACIÓN ⚠

FOR2

Para cada tupla del Cursor
c_matriculas('A004') hacer
 si (feb_jun <> null) entonces
#presentados ++
 si (feb_jun >= 5) entonces
#aprobados ++



f_si
f_si
f_para

← **CURS**OR **C_MATRÍCULA**('A004')

si (**#presentados** > 0) entonces
porcentaje = (**#aprobados** / **#presentados**) * 100;
 si no
porcentaje = 0;
 f_si;
 mostrar_línea_detalle_por_pantalla()
 f_para

...

Asignatura **#Pres** **#Apro** **%Apro**

Base de Datos | 3 1 33.33%



- ↳ El procedimiento, según se indica en su enunciado, tendrá dos parámetros: p_curso y p_año, que serán del mismo tipo que los campos EI.ASIGNATURA.curso y EI.MATRICULA.año respectivamente, por lo que la definición del procedimiento sería la siguiente:



CREATE OR REPLACE

PROCEDURE porcentaje_aprobados(p_curso EI.ASIGNATURA.curso%type,
p_año EI.MATRICULA.año%type) IS



- ↳ A continuación, en primer lugar, vamos a obtener los idAsig y nombres de las asignaturas correspondientes al curso indicado en el primer parámetro del procedimiento, p_curso. Para ello, definiremos un cursor, c_asignaturas, cuya consulta asociada obtenga, de la tabla EI.ASIGNATURA, el conjunto de idAsig, y sus correspondientes nombres, cuyo curso sea el indicado en el parámetro p_curso :



CURSOR c_asignaturas IS

SELECT idAsig, nombre


FROM EI.ASIGNATURA

WHERE curso = **p_curso**;



- ↳ Para cada asignatura, será necesario obtener los valores de las notas de los alumnos en la convocatoria de feb_jun del año especificado en el parámetro p_año. Para ello, definiremos un cursor, c_matrículas, que tomará como parámetro el idAsig de la asignatura que se esté tratando, y cuya consulta asociada sea la que obtenga, de la tabla EI.MATRICULA, las notas de la asignatura de código idAsig en la convocatoria de feb_jun del año especificado en el parámetro del procedimiento p_año.



1  **CURSOR** `c_matriculas`(`p_idAsig` `EI.MATRICULA.idAsig%type`) **IS**
`SELECT idAsig, feb_jun`
`FROM EI.MATRICULA`
`WHERE año=p_año AND idAsig = p_idAsig;`

← **CORRELACIÓN** 



↳ Se propone **controlar, mediante el uso de excepciones**, el hecho de que no existan alumnos matriculados el año que se especifique en la llamada al procedimiento. Para ello, definiremos la excepción `no_existe_año` en la zona de declaraciones y definiremos un cursor implícito que recupere el número de registros existentes de alumnos matriculados en el año indicado, y almacene el valor obtenido en la variable `filas_año`, de manera que, si `filas_año=0`, se deberá activar la excepción `no_existe_año`. Su tratamiento consistirá, simplemente, en mostrar un mensaje indicando que no existen datos para el año que se especificó en la llamada.

▶  La definición y tratamiento de la excepción sería el siguiente:



1

`filas_año` **INTEGER**;
`no_existe_año` **EXCEPTION**;



Se controla la excepción al principio del Bloque de Sentencias

BEGIN

2

`SELECT COUNT(*)` **INTO** `filas_año` **FROM** `EI.MATRICULA`
`WHERE año = p_año AND curso = p_curso`;
`IF filas_año = 0 THEN RAISE no_existe_año`; **END IF**;

3

EXCEPTION

WHEN `no_existe_año` **THEN**

`dbms_output.put_line('No hay información del curso ' || p_año || ' en la BD!!:O')`;





- ↳ El funcionamiento del cuerpo del procedimiento sería el siguiente:
- ▶ En primer lugar, *definiremos un bucle FOR con el que recorreremos cada una de las asignaturas obtenidas mediante el cursor c_asignaturas.*
 - ↳ Como, para cada *asignatura*, tenemos que calcular el número de alumnos *presentados y aprobados*, necesitaremos definir dos variables *acumuladoras*, num_alumnos_presentados y num_alumnos_aprobados, con las que iremos contabilizando, respectivamente, los alumnos presentados y los que han aprobado.
 - ↳ Estas variables se definirán en la zona de declaraciones y se inicializarán a 0 al comienzo de este bucle externo, para cada una de las asignaturas:

```
...  
1 num_alumnos_presentados INTEGER;  
  num_alumnos_aprobados INTEGER;  
  BEGIN  
  ...  
2  FOR v_asignaturas IN c_asignaturas LOOP  
    num_alumnos_presentados := 0; num_alumnos_aprobados := 0;  
    ...  
  END LOOP;  
  ...
```

- ▶ Dentro del bucle anterior será necesario definir otro bucle para recorrer las tuplas del cursor c_matriculas, mediante el que podremos acceder a las notas de los alumnos en la convocatoria de feb_jun del año especificado en el parámetro del procedimiento p_año.



⇒ En este bucle contabilizaremos los alumnos *presentados*, cuántos de ellos han *aprobado* y, al finalizar su ejecución, calcularemos el *porcentaje de aprobados* en función de dichos valores e imprimiremos por pantalla una línea-detalle con dichos valores junto a la *asignatura a la que corresponden*:

2

```
...  
FOR v_asignaturas IN c_asignaturas LOOP  
    num_alumnos_presentados:=0; num_alumnos_aprobados:=0;  
    FOR v_matriculas IN c_matriculas(v_asignaturas.idAsig) LOOP  
        IF (v_matriculas.feb_jun IS NOT NULL) THEN  
            num_alumnos_presentados := num_alumnos_presentados+1;  
            IF (v_matriculas.feb_jun >= 5) THEN  
                num_alumnos_aprobados := num_alumnos_aprobados+1;  
            END IF;  
        END IF;  
    END LOOP;  
    IF (num_alumnos_presentados <> 0) THEN  
        porcentaje := (num_alumnos_aprobados/num_alumnos_presentados)*100;  
    ELSE  
        porcentaje := 0;  
    END IF;  
    dbms_output.put_line( rpad(v_asignaturas.nombre,40) || ' ' ||  
        rpad(num_alumnos_presentados,6) || ' ' ||  
        rpad(num_alumnos_aprobados,6) || ' ' || rpad(porcentaje || '%',7) );  
END LOOP;  
...
```





➡ Definición final del procedimiento:



Parámetros del procedimiento

```
CREATE OR REPLACE  
PROCEDURE porcentaje_aprobados( p_curso EI.ASIGNATURA.curso%type,  
                                p_año EI.MATRICULA.año%type ) IS
```

1 /* Declaraciones locales */

```
num_alumnos_presentados INTEGER;  
num_alumnos_aprobados INTEGER;  
porcentaje NUMBER(5,2);  
existe_año INTEGER;  
no_existe_año EXCEPTION;
```

i Variable que recoge el valor del cursor implícito de control de excepción

```
CURSOR c_asignaturas IS  
SELECT idAsig, nombre  
FROM EI.ASIGNATURA  
WHERE curso = p_curso;
```

```
CURSOR c_matriculas(p_idAsig EI.MATRICULA.idAsig%type) IS  
SELECT idAsig, feb_jun  
FROM EI.MATRICULA  
WHERE año=p_año AND idAsig=p_idAsig;
```





2

BEGIN /* Bloque de Sentencias */

Control de la excepción

**SELECT** COUNT(*) INTO filas_año **FROM** EI.MATRICULA**WHERE** año = p_año **AND** curso = p_curso;**IF** filas_año = 0 **THEN RAISE** no_existe_año; **END IF**;**FOR** v_asignaturas **IN** c_asignaturas **LOOP**

num_alumnos_presentados:=0; num_alumnos_aprobados:=0;

FOR v_matriculas **IN** c_matriculas(v_asignaturas.idAsig) **LOOP****IF** (v_matriculas.feb_jun **IS NOT NULL**) **THEN**

num_alumnos_presentados := num_alumnos_presentados+1;

IF (v_matriculas.feb_jun >= 5) **THEN**

num_alumnos_aprobados := num_alumnos_aprobados+1;

END IF;**END IF**;**END LOOP**;**IF** (num_alumnos_presentados <> 0) **THEN**

porcentaje := (num_alumnos_aprobados/num_alumnos_presentados)*100;

ELSE

porcentaje := 0;

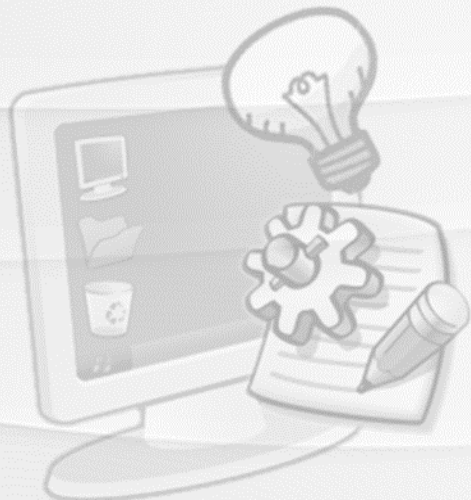
END IF;

dbms_output.put_line(rpad(v_asignaturas.nombre,40) || ' ' ||

rpad(num_alumnos_presentados,6) || ' ' ||

rpad(num_alumnos_aprobados,6) || ' ' || rpad(porcentaje || '%',7));

END LOOP;

**3****/* Bloque de Tratamiento de Excepciones */****EXCEPTION****WHEN no_existe_año THEN****dbms_output.put_line('No hay información del curso ' || p_año || ' en la
BD!!:O');****WHEN OTHERS THEN****dbms_output.put_line('Se produjo un error!!:O');****END porcentaje_aprobados;**

1. PL-SQL – SESIÓN I

➡ **Ejercicio 1.** Diseñar la función `facturacion()`, la cual admite dos parámetros de entrada (un teléfono y un año) y devuelve la facturación total de ese número en ese año.

TARIFA COSTE ↙ ↘ LLAMADA DURACIÓN ↙ LLAMADA TF_ORIGEN ↘ LLAMADA FECHA_HORA

La función debe controlar 2 tipos de excepciones:

- ▶ el teléfono no existe o el teléfono no ha realizado llamadas ese año.
- ▶ la facturación del teléfono es inferior a 1 euro.

Ejemplo de ejecución:

```
SQL> call dbms_output.put_line(facturacion('654123321', 2006));
```

10,18

Llamada terminada.





Implementación de la función



Definimos la estructura general de nuestra función:

```
CREATE or REPLACE  
FUNCTION facturacion (p_tf_origen MF.LLAMADA.tf_origen%type, p_año INTEGER)  
RETURN Float IS
```

← **VALOR RETORNADO**

← **PARÁMETROS**

- 1** /* Declaraciones locales */
BEGIN
- 2** /* Sentencias */
EXCEPTION
- 3** /* Tratamiento de Excepciones */
END facturación;



2

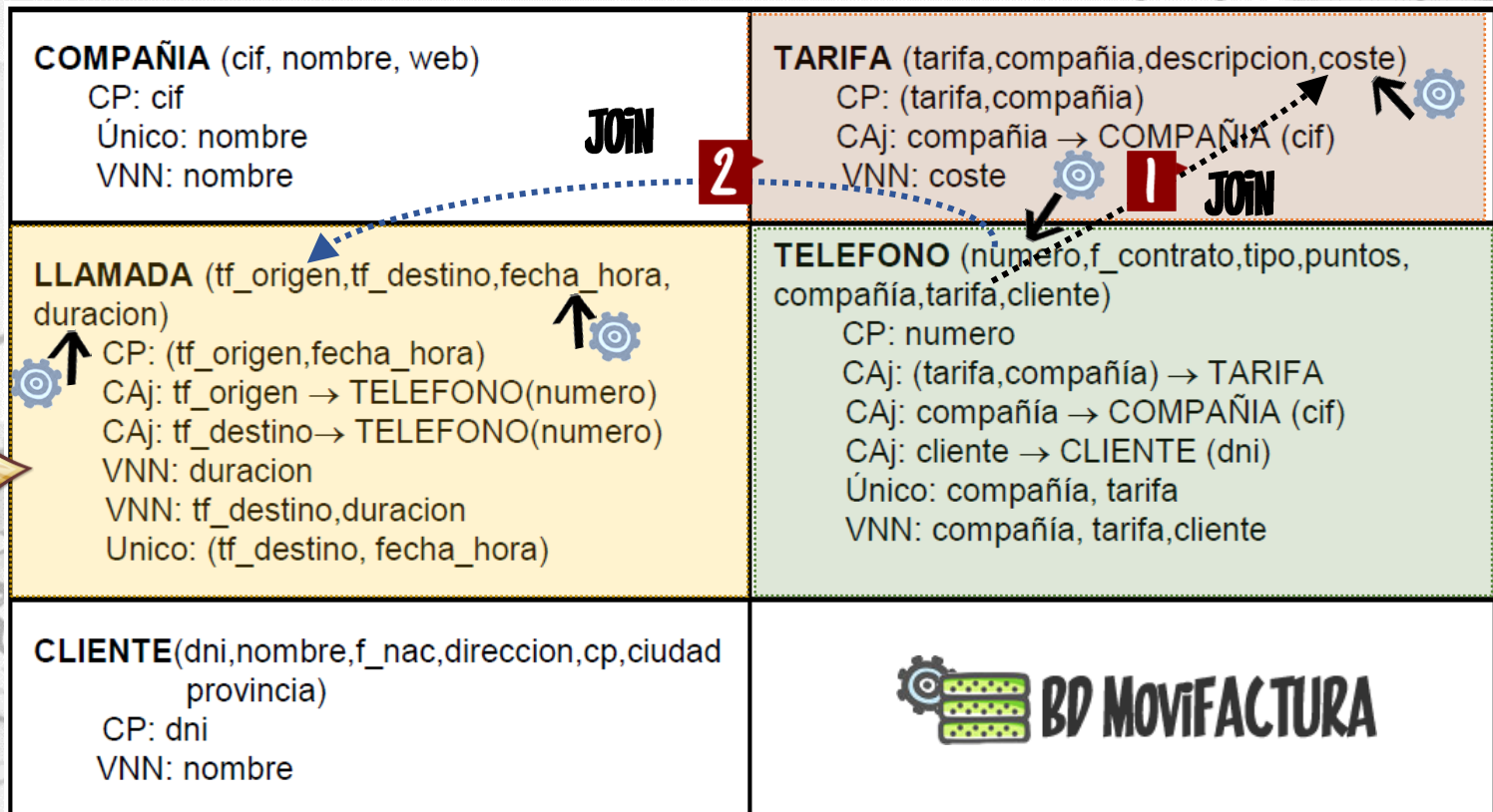


Observando la estructura de la BD y los datos que hemos de manejar en la consulta, se deduce que será necesario realizar un INNER JOIN de las tablas TELEFONO, LLAMADA y TARIFA.



A continuación, seleccionaremos, del resultado del anterior INNER JOIN, los datos de facturación correspondientes al *teléfono* y *año* recibidos como *parámetros*.

Para calcular los datos de facturación totales de cada teléfono tendremos que calcular la suma de los importes de cada una de las llamadas realizadas: $\Sigma((\text{llam.duración} * \text{tar.coste}) / 60)$, para lo cual, agruparemos los datos de las llamadas por llam.tel origen.





Para calcular los *importes totales* asociados a cada teléfono será necesario agrupar por TF_ORIGEN. La versión inicial de nuestra consulta sería la siguiente:

Consulta previa...

```

1 SELECT LLAM.tf_origen, TRUNC(SUM((LLAM.duracion*TAR.coste)/60),2) Importe_Total
2 FROM ( MF.Telefono TEL INNER JOIN MF.Tarifa TAR USING(tarifa, compañia))
3      INNER JOIN MF.Llamada LLAM ON LLAM.tf_origen=TEL.numero
4 GROUP BY LLAM.tf_origen;

```

Hacemos el INNER JOIN de las tablas que participan en la consulta, y agrupamos por número de teléfono origen

Resultado de la Consulta x

Todas las Filas Recuperadas: 8 en 0,038 segundos

	TF_ORIGEN	IMPORTE_TOTAL
1	678234234	11,96
2	654345345	4,11
3	654123321	10,18
4	678111222	4,02
5	654789789	4,62
6	654012012	3,77
7	666789789	2,14
8	666010101	2,43

Importe total de cada teléfono durante todo el tiempo de contrato



↳ La consulta deberá devolver un único valor, el correspondiente al importe total asociado al teléfono y año especificado, que lo almacenaremos en la variable `Importe_Total` de un cursor implícito (`SELECT...INTO Importe_Total`), variable que definiremos previamente en la sección de declaraciones de tipo FLOAT.

↳ A continuación, tendremos que parametrizar la consulta anterior en base a los parámetros de entrada de nuestra función, de manera que ya podremos establecer el criterio de selección basado en el número de teléfono y el año indicado en la llamada a la función. Por tanto, el bloque de la función sería el siguiente:

```
SELECT TRUNC(SUM((LLAM.duracion*TAR.coste)/60),2) INTO Importe_Total
FROM ( MF.Telefono TEL INNER JOIN MF.Tarifa TAR USING(tarifa, compañía))
      INNER JOIN MF.Llamada LLAM ON LLAM.tf_origen=TEL.numero
WHERE EXTRACT (YEAR FROM LLAM.fecha_hora) = p_año
      AND LLAM.tf_origen = p_tf_origen
GROUP BY LLAM.tf_origen;
```



3

↳ Por último, tendríamos que tratar las excepciones que pudieran darse durante la ejecución de la función:

- ▶ Que el teléfono no exista o el teléfono no haya realizado llamadas ese año lo controlaremos mediante la excepción `NO_DATA_FOUND` (Oracle - la consulta no devuelve ningún valor).
- ▶ Que la facturación del teléfono sea inferior a 1 euro, lo controlaremos mediante una excepción de usuario, `facturacionBaja`, que tendremos que declarar en la sección de declaraciones y activarla cuando el valor obtenido en la consulta, que estará almacenado en la variable `Importe_Total`, tiene un valor inferior a 1.

**3**

La excepción facturacionBaja se te verificará tras la consulta de la siguiente manera:

```
IF (Importe_Total < 1) THEN  
    RAISE facturacionBaja;  
END IF;
```

Activamos la excepción si Importe_total es inferior a 1€

Una vez verificado que no se haya producido una facturación inferior a 1 que hubiera activado la excepción correspondiente, se hace el RETURN de la función. Se retornará como resultado el valor de Importe_Total obtenido por la consulta

```
RETURN Importe_Total ;
```

Por último, tendremos que tratar las distintas excepciones en la correspondiente sección EXCEPTION:

EXCEPTION

WHEN facturacionBaja THEN

```
dbms_output.put_line('Facturacion demasiado baja!! :O');  
RETURN -1;
```

WHEN NO_DATA_FOUND THEN

```
dbms_output.put_line('El teléfono no existe o no ha realizado llamadas ese año');  
RETURN -1;
```

WHEN OTHERS THEN

```
dbms_output.put_line('Ha ocurrido un error!! :(');  
RETURN -1;
```





Definición final de nuestra función:

Parámetros de la función



```
CREATE or REPLACE
FUNCTION facturacion ( p_tf_origen mf.llamada.tf_origen%type, p_año integer)
```

⚠ RETURN FLOAT IS

1 /* Declaraciones locales */

```
Importe_Total NUMBER(10, 2);
facturacionBaja EXCEPTION;
```

BEGIN

2 /* Bloque de Sentencias */

```
SELECT TRUNC(SUM((LLAM.duracion*TAR.coste)/60),2) INTO Importe_Total
FROM ( MF.Telefono TEL INNER JOIN MF.Tarifa TAR USING(tarifa, compañía))
INNER JOIN MF.Llamada LLAM ON LLAM.tf_origen=TEL.numero
WHERE EXTRACT (YEAR FROM LLAM.fecha_hora) = p_año
AND LLAM.tf_origen = p_tf_origen
GROUP BY LLAM.tf_origen;
```

❏ Consulta que obtiene el importe total del un teléfono en un año concreto, siendo estos últimos, los valores recibidos como parámetros de la función

⚠ Parámetros de la función

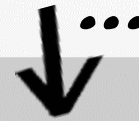
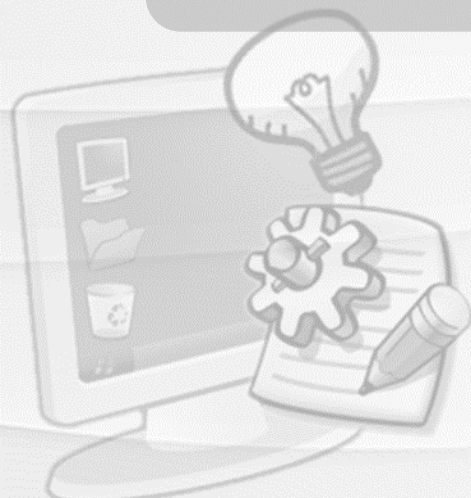
```
IF (Importe_Total < 1) THEN
RAISE facturacionBaja;
END IF;
```

❏ Activamos la excepción si Importe_total es inferior a 1

⚠ RETURN Importe_Total;

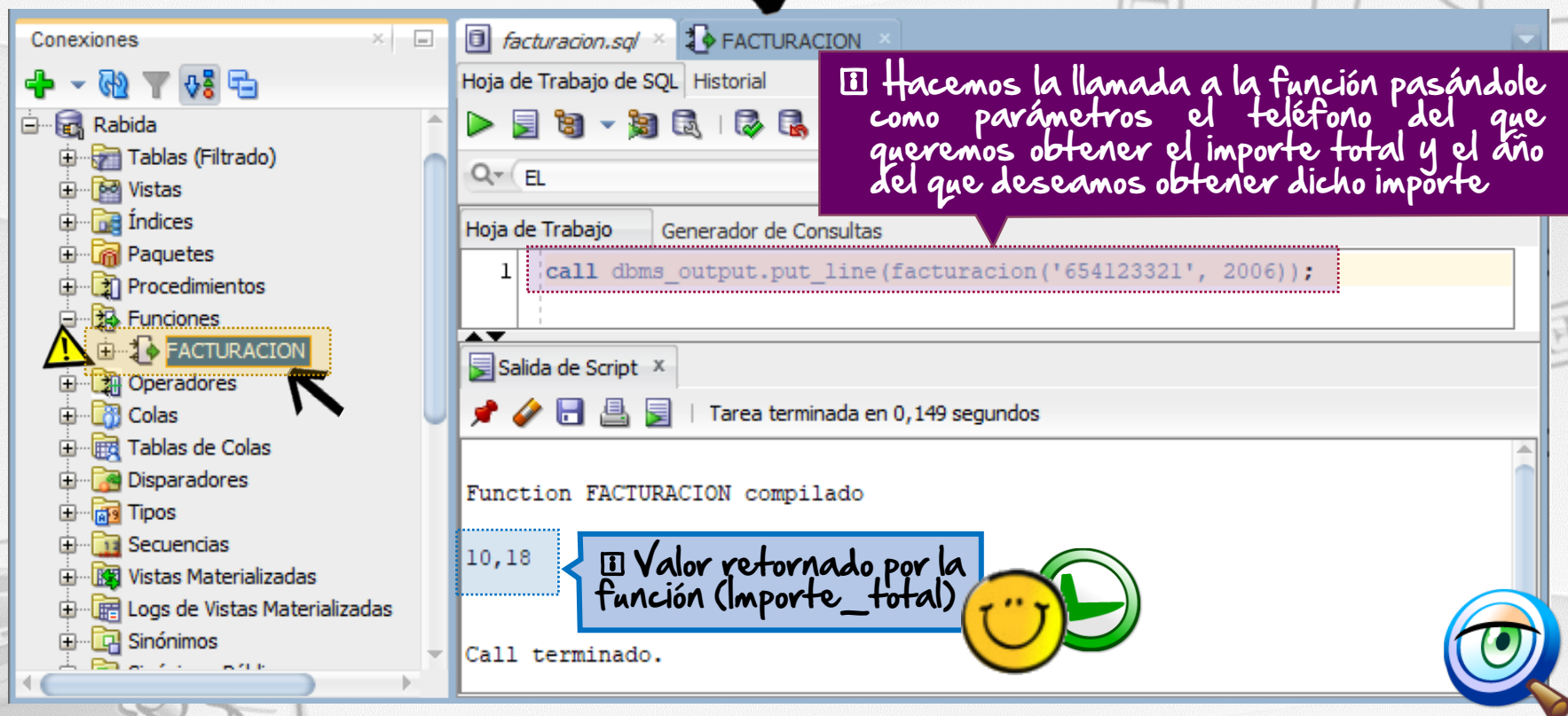
❏ Retornamos el valor de la función (Importe_total)



**3****/* Tratamiento de Excepciones*/****EXCEPTION****WHEN facturacionBaja THEN****dbms_output.put_line('Facturacion demasiado baja!! :O');****RETURN -1;****WHEN NO_DATA_FOUND then****dbms_output.put_line('El teléfono no existe o no ha realizado llamadas ese año');****RETURN -1;****WHEN others then****dbms_output.put_line('Ha ocurrido un error!! :(');****RETURN -1;****END facturacion;**



Ejemplo de llamada a nuestra función:



Conexiones

- Rabida
 - Tablas (Filtrado)
 - Vistas
 - Índices
 - Paquetes
 - Procedimientos
 - Funciones
 - FACTURACION**
 - Operadores
 - Colas
 - Tablas de Colas
 - Disparadores
 - Tipos
 - Secuencias
 - Vistas Materializadas
 - Logs de Vistas Materializadas
 - Sinónimos

facturacion.sql x **FACTURACION** x

Hoja de Trabajo de SQL Historial

Q EL

Hoja de Trabajo Generador de Consultas

1 `call dbms_output.put_line(facturacion('654123321', 2006));`

Salida de Script x

Tarea terminada en 0,149 segundos




Function FACTURACION compilado

10,18

Call terminado.

Annotations:

- 1** Hacemos la llamada a la función pasándole como parámetros el teléfono del que queremos obtener el importe total y el año del que deseamos obtener dicho importe
- 1** Valor retornado por la función (Importe_total)





1. PL-SQL – SESIÓN I

➡ **Ejercicio 2.** Diseñar el procedimiento `LlamadaFacturacion(año)`, el cual, para cada teléfono de la tabla `LLAMADA`, debe realizar una llamada a la función `facturación(tel,año)` y mostrar la facturación de dicho teléfono en el año que se le pase como parámetro.

Ejemplo de ejecución:

```
SQL> execute llamadaFacturacion(2006);
```

Nº teléfono Importe (en €)

654012012	3,78
654123321	10,18
654345345	4,11
654789789	4,63
666010101	2,44
666789789	2,14
678111222	4,02
678234234	11,96

Valores obtenidos al aplicar la función `facturación(año)` a cada uno de los teléfonos obtenidos en el cursor `c_teléfonos` en el año especificado en la llamada

Lista de teléfonos que realizaron llamadas en 2006

CURSOR c_telefonos

Procedimiento PL/SQL terminado correctamente. 10,18








Implementación del procedimiento

 Definimos la estructura general de nuestro procedimiento:

```
CREATE or REPLACE  
PROCEDURE LlamadaFacturacion(p_año INTEGER) IS
```

↑  **PARÁMETRO**

- 1**  **/* Declaraciones locales */**
BEGIN
- 2**  **/* Bloque de Sentencias */**
EXCEPTION
- 3**  **/* Tratamiento de Excepciones */**
END LlamadaFacturacion;





Como se nos pide mostrar la facturación de cada teléfono de la tabla LLAMADAS en el año que se le pasa como parámetro, lo primero que haremos será definir en la *zona de declaraciones locales* un cursor, *c* telefonos, cuya consulta asociada deberá devolver el conjunto de teléfonos desde los que se realizaron llamadas (tf_origen) en el año que reciba como parámetro de entrada del procedimiento:

```
CREATE OR REPLACE PROCEDURE LlamadaFacturacion(p_año INTEGER) IS
```



```
CURSOR c_telefonos IS
```

```
SELECT DISTINCT tf_origen FROM MF.LLAMADA  
WHERE EXTRACT (YEAR FROM fecha_hora) = p_año
```



A continuación, en el *Bloque de Sentencias*, utilizaremos un *bucle* para recorrer cada una de las tuplas recuperadas por el cursor y, para cada teléfono de dichas tuplas, realizaremos una llamada a la función *facturación*(tel,año), definida en el ejercicio anterior, mediante la que calcularemos cuál fue su facturación en el año especificado. Se mostrará una línea de detalle por cada tupla formada por el teléfono y su facturación tal como se mostró en la ejecución de ejemplo el enunciado.

2

El bucle podría ser el siguiente:



```
FOR r_telefono IN c_telefonos LOOP
```

```
  dbms_output.put_line( r_telefono.tf_origen || ' ' ||
```

```
    facturacion(r_telefono.tf_origen, p_año));
```

```
END LOOP;
```





Rabida x

Hoja de Trabajo | Generador de Consultas

```

1 select distinct tf_origen
2 from MF.LLAMADA
3 where extract (year from fecha_hora) = 2006;
    
```

Resultado de la Consulta x

Todas las Filas Recuperadas: 8 en 0,168 segundos

TF_ORIGEN
1 654012012
2 654123321
3 654345345
4 654789789
5 666010101
6 666789789
7 678111222
8 678234234

C_TELEFONOS(2006)

TF_ORIGEN

→ 1 654012012
→ 2 654123321
→ 3 654345345
→ 4 654789789
... 5 666010101
• 6 666789789
• 7 678111222
→ 8 678234234

facturacion(654012012,2006) (=3,78)

Nº teléfono	Importe (en €)
654012012	3,78
...	...

3 Por último, tendríamos que tratar las excepciones que pudieran darse durante la ejecución de la función. En este caso, como no se especifica ninguna circunstancia particular que debiera contemplarse, trataremos las excepciones de manera genérica:



... EXCEPTION

WHEN OTHERS THEN

dbms_output.put_line('Ha ocurrido un error!!(' ');

...

Podíamos documentar el error mediante SQLCODE y SQLERRM:
 DBMS_OUTPUT.PUT_LINE('Error' || SQLCODE || '-' || SQLERRM);





Definición final de nuestro procedimiento:

Parámetro del procedimiento



```
CREATE OR REPLACE PROCEDURE LlamadaFacturacion(p_año INTEGER) IS
```

```
/* Declaraciones locales */
```

1

```
CURSOR c_telefonos IS  
SELECT DISTINCT tf_origen FROM MF.LLAMADA  
WHERE EXTRACT (YEAR FROM fecha_hora) = p_año
```

1 Cursor cuya consulta que obtiene todos los teléfonos desde los que se realizaron llamadas en un año concreto, siendo éste, el valor recibido como parámetro de entrada del procedimiento

```
/* Bloque de Sentencias */
```

2

```
BEGIN  
dbms_output.put_line('Nº teléfono' || ' ' || 'Importe (en €)');  
dbms_output.put_line('-----');
```

FOR r_telefono **IN** c_telefonos **LOOP**
dbms_output.put_line(r_telefono.tf_origen || ' ' || facturacion(r_telefono.tf_origen, **p_año**));
END LOOP;

1 Mostramos por pantalla los teléfonos y sus totales de facturación para el año indicado

1 Llamada a la función facturación para el teléfono que estemos tratando en el cursor


```
/* Tratamiento de Excepciones */
```


3

```
EXCEPTION  
WHEN others THEN  
dbms_output.put_line('Ha ocurrido un error!!:');  
END;
```

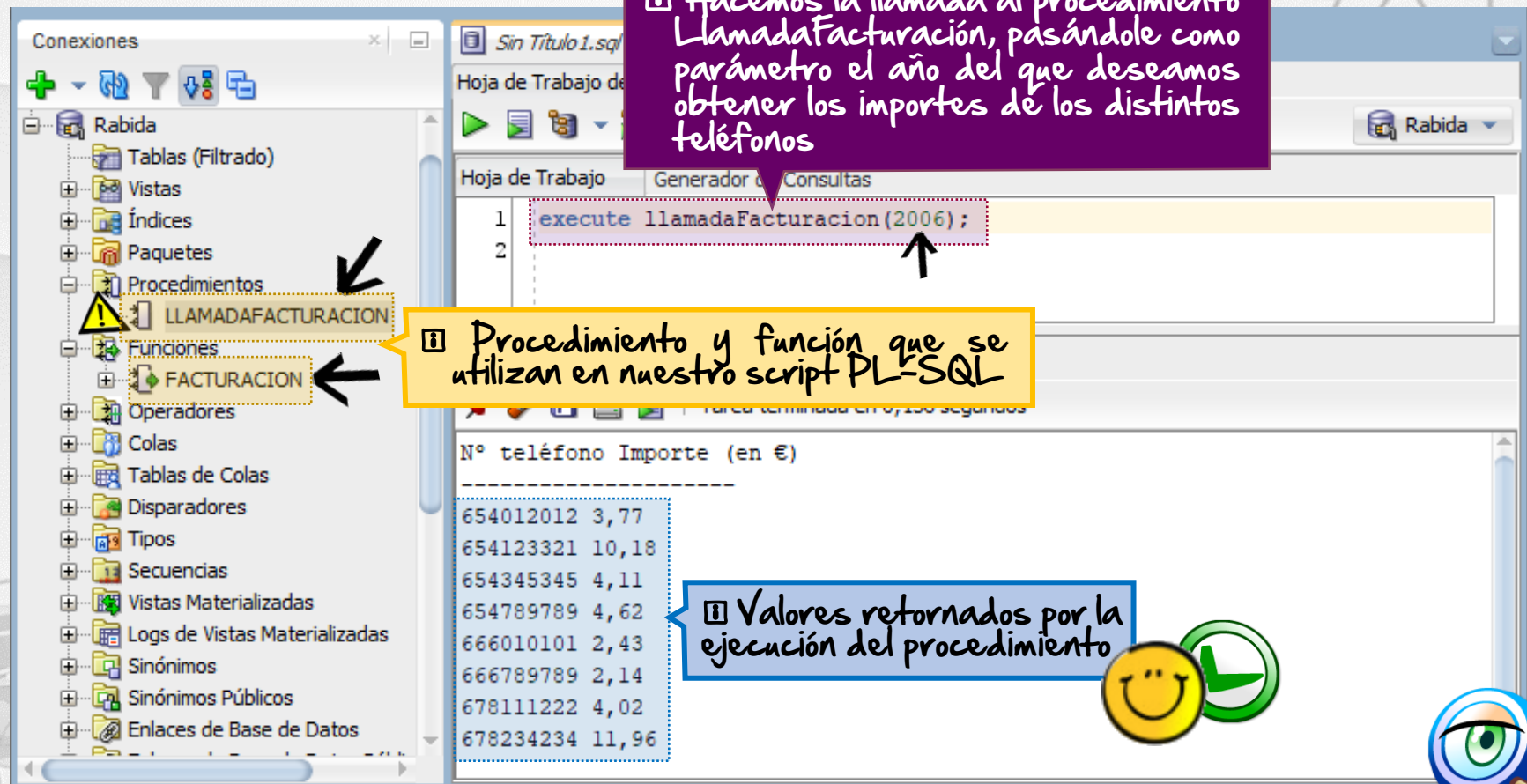
1 Capturamos las excepciones que se produzcan

Ejemplo de ejecución de nuestro procedimiento

 Hacemos la llamada al procedimiento `LlamadaFacturación`, pasándole como parámetro el año del que deseamos obtener los importes de los distintos teléfonos

 Procedimiento y función que se utilizan en nuestro script PL-SQL

 Valores retornados por la ejecución del procedimiento



The screenshot shows the SQL Developer interface with the 'Conexiones' pane on the left, the 'Sin Título1.sql' script in the center, and the 'Generador de Consultas' pane on the right. The 'Conexiones' pane shows the 'Rabida' database with various objects. The 'Sin Título1.sql' script contains the following code:

```
1 execute llamadaFacturacion(2006);
2
```

The 'Generador de Consultas' pane shows the results of the execution, displaying a table with the following data:

Nº teléfono	Importe (en €)
654012012	3,77
654123321	10,18
654345345	4,11
654789789	4,62
666010101	2,43
666789789	2,14
678111222	4,02
678234234	11,96

Annotations in the image include:

- A purple box pointing to the procedure name in the script: "Hacemos la llamada al procedimiento `LlamadaFacturación`, pasándole como parámetro el año del que deseamos obtener los importes de los distintos teléfonos".
- A yellow box pointing to the procedure and function in the 'Conexiones' pane: "Procedimiento y función que se utilizan en nuestro script PL-SQL".
- A blue box pointing to the results table: "Valores retornados por la ejecución del procedimiento".

Decorative elements include a smiley face, a green clock, and a magnifying glass icon.