

Bases de Datos 1

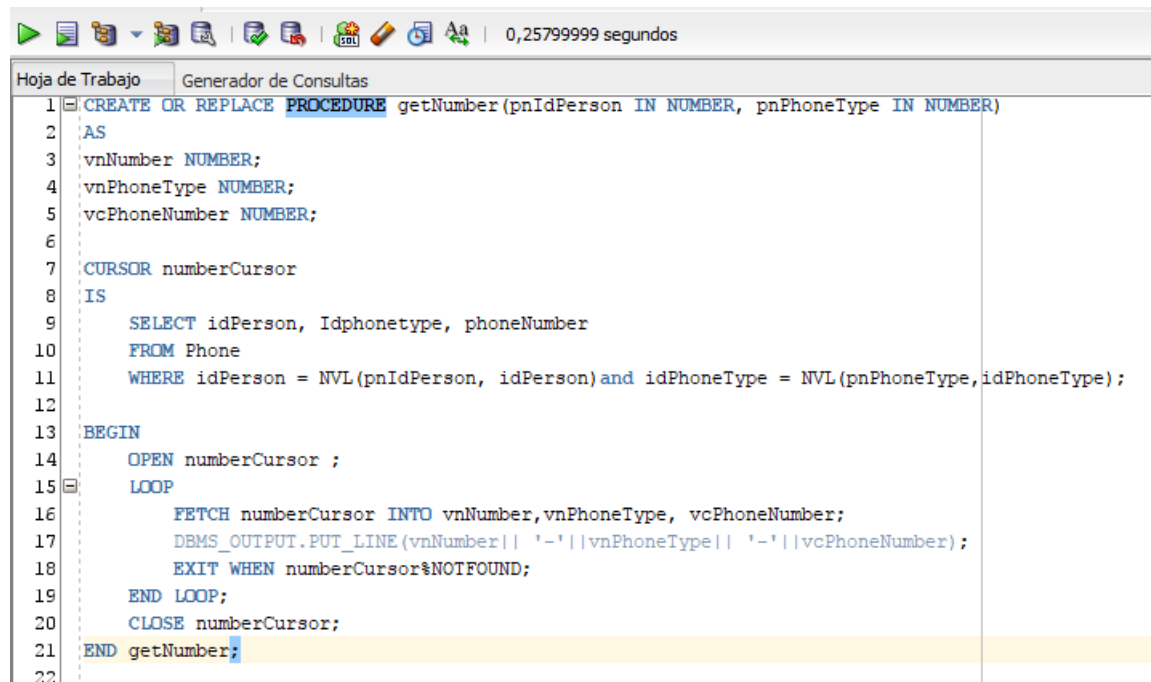
Laboratorio 6

- Fabián Bustos
- Ian Murillo

Evidencia Laboratorio 6

Ejercicio 1:

Cree un procedimiento (utilizando OPEN del cursor) que retorne todos los teléfonos de una persona. El número de persona debe ser enviado como parámetro así como el tipo de teléfono y este último puede ser nulo, en cuyo caso se deben desplegar todos los teléfonos de la persona.

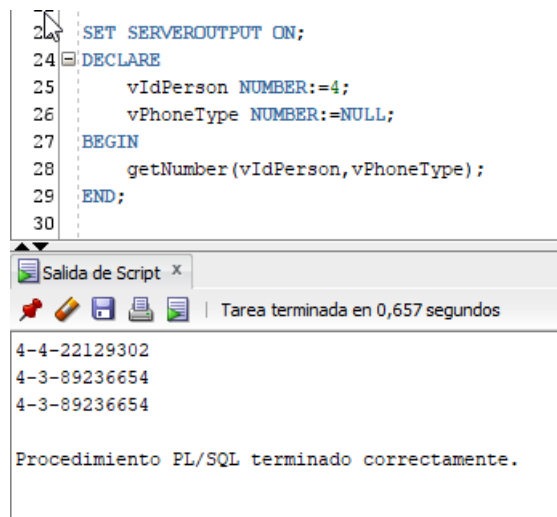


```
1 CREATE OR REPLACE PROCEDURE getNumber(pnIdPerson IN NUMBER, pnPhoneType IN NUMBER)
2 AS
3   vnNumber NUMBER;
4   vnPhoneType NUMBER;
5   vcPhoneNumber NUMBER;
6
7   CURSOR numberCursor
8   IS
9     SELECT idPerson, Idphonetype, phoneNumber
10    FROM Phone
11   WHERE idPerson = NVL(pnIdPerson, idPerson) and idPhoneType = NVL(pnPhoneType, idPhoneType);
12
13 BEGIN
14   OPEN numberCursor ;
15   LOOP
16     FETCH numberCursor INTO vnNumber, vnPhoneType, vcPhoneNumber;
17     DBMS_OUTPUT.PUT_LINE(vnNumber || '-' || vnPhoneType || '-' || vcPhoneNumber);
18     EXIT WHEN numberCursor%NOTFOUND;
19   END LOOP;
20   CLOSE numberCursor;
21 END getNumber;
```

Ejercicio 2:

Cree un test para probar el procedimiento utilizando cursores.

Mediante el siguiente Test se puede observar el funcionamiento del cursor:



The screenshot shows a SQL Developer script editor with the following code:

```
23 SET SERVEROUTPUT ON;  
24 DECLARE  
25     vIdPerson NUMBER:=4;  
26     vPhoneType NUMBER:=NULL;  
27 BEGIN  
28     getNumber(vIdPerson,vPhoneType);  
29 END;  
30
```

Below the script editor is a window titled "Salida de Script x" showing the output of the script:

```
Tarea terminada en 0,657 segundos  
  
4-4-22129302  
4-3-89236654  
4-3-89236654  
  
Procedimiento PL/SQL terminado correctamente.
```

Ejercicio 3:

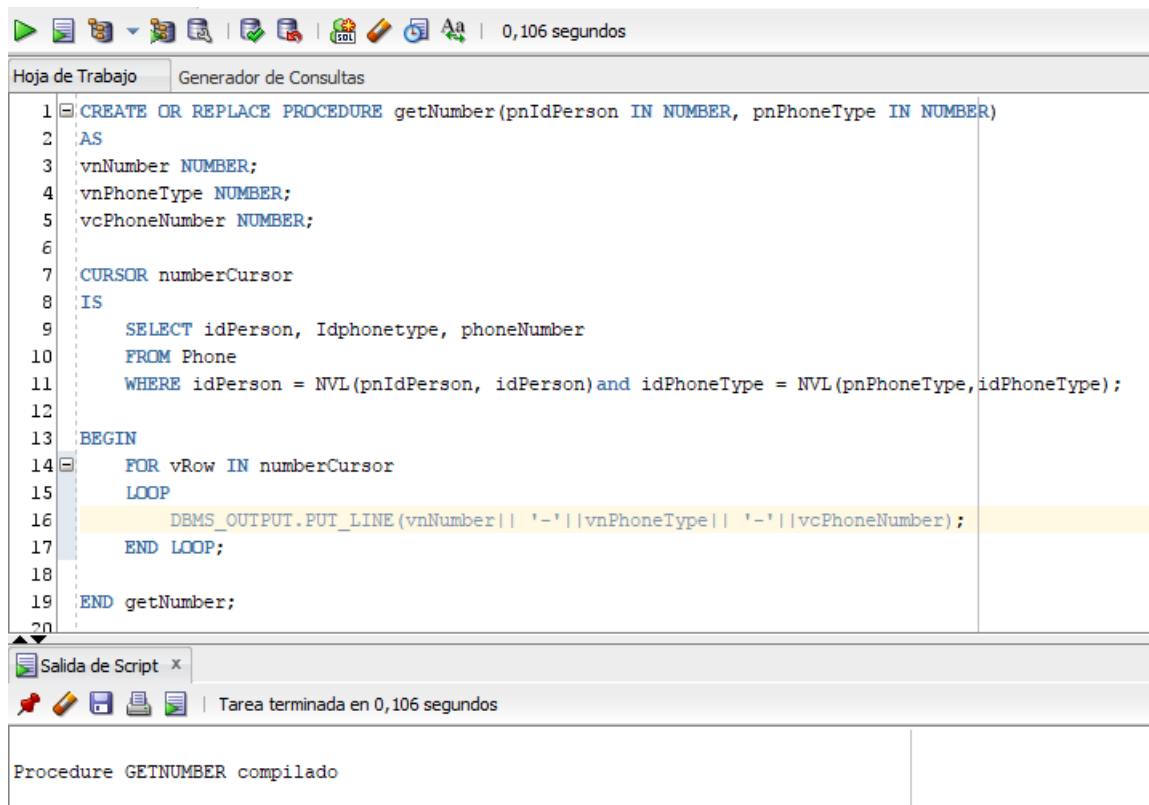
¿Qué pasa cuando en el fetch del test se agregan más campos?

Ejercicio 4:

Ejercicio 5:

Ejercicio 6:

Modifique el cursor para utilizar el código con LOOP.



```
1 CREATE OR REPLACE PROCEDURE getNumber(pnIdPerson IN NUMBER, pnPhoneType IN NUMBER)
2 AS
3   vnNumber NUMBER;
4   vnPhoneType NUMBER;
5   vcPhoneNumber NUMBER;
6
7   CURSOR numberCursor
8   IS
9     SELECT idPerson, Idphonetype, phoneNumber
10    FROM Phone
11   WHERE idPerson = NVL(pnIdPerson, idPerson) and idPhoneType = NVL(pnPhoneType, idPhoneType);
12
13 BEGIN
14   FOR vRow IN numberCursor
15   LOOP
16     DBMS_OUTPUT.PUT_LINE(vnNumber || '-' || vnPhoneType || '-' || vcPhoneNumber);
17   END LOOP;
18
19 END getNumber;
```

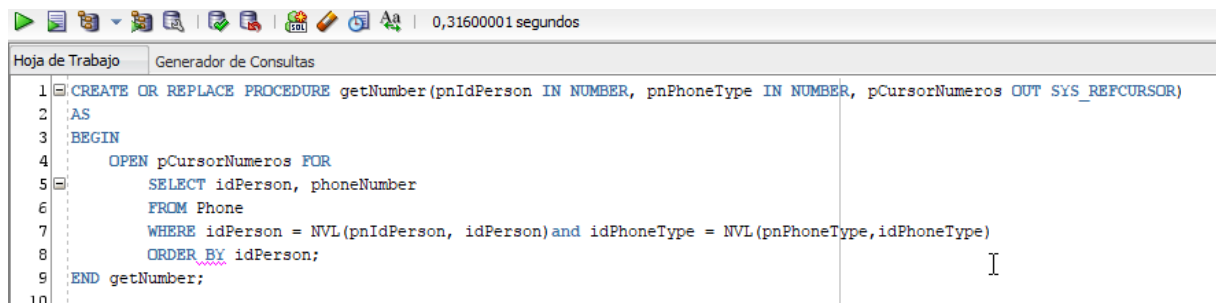
Salida de Script x

Tarea terminada en 0,106 segundos

Procedure GETNUMBER compilado

Ejercicio 7:

Modifique el cursor para utilizar el código de SYS_REFCURSOR.



```
1 CREATE OR REPLACE PROCEDURE getNumber(pnIdPerson IN NUMBER, pnPhoneType IN NUMBER, pCursorNumeros OUT SYS_REFCURSOR)
2 AS
3 BEGIN
4   OPEN pCursorNumeros FOR
5     SELECT idPerson, phoneNumber
6    FROM Phone
7   WHERE idPerson = NVL(pnIdPerson, idPerson) and idPhoneType = NVL(pnPhoneType, idPhoneType)
8   ORDER BY idPerson;
9 END getNumber;
```

Se prueba que funciona con el siguiente test:

```
11 DECLARE
12
13 pnIdPerson NUMBER :=3;
14 pnPhoneType NUMBER :=NULL;
15 pCursorNumeros SYS_REFCURSOR;
16 vidPerson NUMBER;
17 vphoneNumber NUMBER;
18 BEGIN
19     getNumber(pnIdPerson,pnPhoneType,pCursorNumeros);
20     LOOP
21         FETCH pCursorNumeros
22         INTO vidPerson, vphoneNumber;
23         EXIT WHEN pCursorNumeros%NOTFOUND;
24         DBMS_OUTPUT.PUT_LINE(vidPerson|| '-'||vphoneNumber);
25     END LOOP;
26     CLOSE pCursorNumeros;
27 END;
```

Salida de Script x

Tarea terminada en 0,316 segundos

3-88329482
3-22782932
3-22329482

Procedimiento PL/SQL terminado correctamente.

Ejercicio 8:

Cree el esquema UN y cree las tablas estudiante, cursos y sus asociaciones.

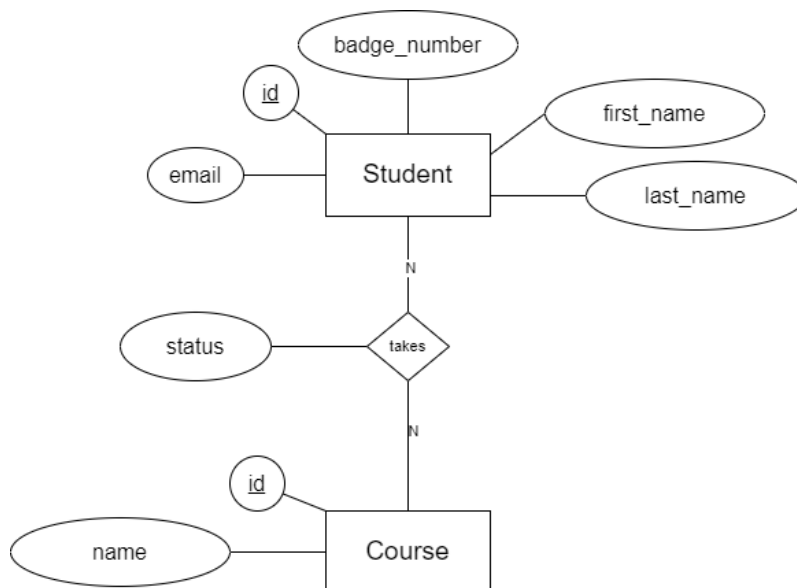
Cree un cursor getCursos que retorne los cursos de un estudiante a partir del id_estudiante y del estado del curso (ya sea aprobado, reprobado, pendiente) donde este último parámetro puede ser nulo en cuyo caso debe retornar todos los cursos del estudiante y su estado.

```

1 create or replace PROCEDURE getCourses(pnIdStudent IN NUMBER, pnCourseStatus IN VARCHAR2)
2 AS
3     vFirst VARCHAR2(30);
4     vLast VARCHAR2(30);
5     vStatus VARCHAR2(30);
6     cName VARCHAR2(30);
7     CURSOR obtenerCursos(pnIdStudent IN NUMBER, pnCourseStatus IN VARCHAR2)
8 IS
9     SELECT p.first_name, p.last_name, r.course_status, c.course_name
10    INTO vFirst, vLast, vStatus, cName
11   FROM student p
12  INNER JOIN studentxcourse r ON p.id_student = r.id_student
13  INNER JOIN course c ON r.id_course = c.id_course
14  WHERE r.course_status = NVL(pnCourseStatus, r.course_status) AND p.id_student = pnIdStudent;
15
16 BEGIN
17     FOR i IN obtenerCursos LOOP
18         DBMS_OUTPUT.PUT_LINE(i.vFirst);
19         DBMS_OUTPUT.PUT_LINE(i.vLast);
20         DBMS_OUTPUT.PUT_LINE(i.vStatus);
21         DBMS_OUTPUT.PUT_LINE(i.cName);
22     END LOOP;
23 END getCourses;
24
25
26
27
28
29

```

Los modelos lógico y conceptual del punto 8 son los siguientes:



Student	
PK	<u>id_student</u>
	first_name
	last_name
	email
	badge_number

Course	
PK	<u>id_course</u>
	id_course
	name

StudentXCourse	
PK	<u>id_course</u>
FK	id_student
FK	id_course
	status