**Prueba técnica PL/SQL**

**Apartado 1:** crear la definición de las tablas, teniendo en cuenta las siguientes especificaciones:

* Una tabla se denominará "PROVINCIAS". Debe tener como mínimo un campo alfanumérico que será único y que identifique el registro, un campo con la descripción y un campo que indique si tiene una baja lógica o está vigente el registro.

CREATE TABLE PROVINCIAS (

--He utilizado de referencia para el tamaño de la id los codigos que da el INE

--https://www.ine.es/daco/daco42/codmun/codmun02/02codmun.xls

id\_provincia VARCHAR(2) PRIMARY KEY,

descripcion VARCHAR(50) NOT NULL,

f\_baja DATE DEFAULT NULL

);

* Otra tabla se denominará "MUNICIPIOS”. Debe contener como mínimo un campo alfanumérico que será único y que identifique el registro, un campo con la descripción, un campo que identifique a que provincia pertenece el municipio (debe existir en la tabla “PROVINCIAS”), un campo numérico con el número de habitantes y un campo que indique si tiene una baja lógica o está vigente el registro.

CREATE TABLE MUNICIPIOS (

id\_municipio VARCHAR(4) PRIMARY KEY,

descripcion VARCHAR(50) NOT NULL,

num\_habitantes NUMBER,

f\_baja DATE DEFAULT NULL,

id\_provincia VARCHAR2(2) NOT NULL,

CONSTRAINT fk\_municipio\_provincia FOREIGN KEY (id\_provincia) REFERENCES PROVINCIAS(id\_provincia)

);

* Por último, una tabla se denominará "USUARIOS" y debe contener como mínimo un campo alfanumérico que sea el DNI. Este campo será único. También tendrá un campo para cada uno de los siguientes datos: nombre, apellidos, teléfono, dirección, un campo que identifique a que municipio pertenece la dirección y un campo que indique si tiene una baja lógica o está vigente el registro.

CREATE TABLE USUARIOS (

dni VARCHAR(9) PRIMARY KEY,

nombre VARCHAR(50) NOT NULL,

apellidos VARCHAR(100) NOT NULL,

telefono VARCHAR(15) NOT NULL,

direccion VARCHAR(200) NOT NULL,

num\_habitantes NUMBER,

f\_baja DATE DEFAULT NULL,

id\_municipio VARCHAR2(4) NOT NULL,

CONSTRAINT fk\_usuario\_municipio FOREIGN KEY (id\_municipio) REFERENCES MUNICIPIOS(id\_municipio)

);

**Apartado 2:** ¿Cuál sería la instrucción de consulta para recuperar el DNI, nombre y apellidos de todos los usuarios de una misma población, por ejemplo = ‘Valencia’? Tener en cuenta que no se quieren obtener los datos con baja lógica. Se valorará si se puede indicar la respuesta de dos formas: con un cursor implícito y con un cursor explícito.

--CURSOR EXPLICITO

DECLARE

v\_ciudad municipios.descripcion%type := &n\_ciudad; --'Valencia'

CURSOR c\_usuarios\_vlc IS

SELECT u.dni, u.nombre, u.apellidos

FROM USUARIOS u

JOIN MUNICIPIOS m using(id\_municipio)

WHERE m.descripcion = v\_ciudad

AND u.f\_baja IS NULL;

v\_dni usuarios.dni%type;

v\_nombre usuarios.nombre%type;

v\_apellidos usuarios.apellidos%type;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Usuarios de ' || v\_ciudad || ':');

OPEN c\_usuarios\_vlc;

LOOP

FETCH c\_usuarios\_vlc into v\_dni, v\_nombre, v\_apellidos;

EXIT WHEN c\_usuarios\_vlc%notfound;

DBMS\_OUTPUT.PUT\_LINE('DNI: ' || v\_dni || ', Nombre: ' || v\_nombre || ', Apellidos: ' || v\_apellidos);

END LOOP;

CLOSE c\_usuarios\_vlc;

END;

-- NO ENTIENDO A QUE OS REFERIS CON CURSOR IMPLICITO

-- ESTO VA A DEVOLVER TOO\_MANY\_ROWS Y NO SERVIRIA

-- CURSOR IMPLICITO

DECLARE

v\_ciudad municipios.descripcion%type := &n\_ciudad; --'Valencia'

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Usuarios de ' || v\_ciudad || ':');

SELECT u.dni, u.nombre, u.apellidos

INTO v\_dni, v\_nombre, v\_apellidos

FROM USUARIOS u

JOIN MUNICIPIOS m using(id\_municipio)

WHERE m.descripcion = v\_ciudad

AND u.f\_baja IS NULL;

DBMS\_OUTPUT.PUT\_LINE('DNI: ' || usuario.dni || ', Nombre: ' || usuario.nombre || ', Apellidos: ' || usuario.apellidos);

END LOOP;

END;

**Apartado 3:** Crear la definición y el cuerpo del paquete PCK\_EJERCICIO\_PRUEBA que contendrá la función F\_Municipios\_Validos. Dicha función tendrá como parámetro de entrada (P\_cod\_municipio). La función buscará en la tabla “MUNICIPIOS” por el código único que identifica cada municipio. Devolverá TRUE si se cumple:

* Cuando el número de habitantes >= 1000, indistintamente de la provincia y del indicador de baja.
* Si no está de baja, la provincia = ‘12’ y el número de habitantes >= 400 y < 1000.
* Si no está de baja, la provincia = ‘46’ y el número de habitantes >= 650 y < 1000.
* Si no está de baja, la provincia = ‘03’ y el número de habitantes > 500 y < 1000.

En el resto de casos, la función devolverá FALSE.

CREATE OR REPLACE PACKAGE PCK\_EJERCICIO\_PRUEBA AS

FUNCTION f\_Municipios\_Validos (p\_cod\_municipio IN MUNICIPIOS.id\_municipio%type) RETURN BOOLEAN;

END PCK\_EJERCICIO\_PRUEBA;

/

CREATE OR REPLACE PACKAGE BODY PCK\_EJERCICIO\_PRUEBA AS

FUNCTION f\_Municipios\_Validos (p\_cod\_municipio IN MUNICIPIOS.id\_municipio%type)

RETURN BOOLEAN

AS

v\_municipio municipios%rowtype;

BEGIN

SELECT \*

INTO v\_municipio

FROM MUNICIPIOS

WHERE id\_municipio = p\_cod\_municipio;

IF v\_municipio.num\_habitantes >= 1000 THEN

RETURN TRUE;

ELSIF v\_municipio.f\_baja IS NOT NULL THEN

RETURN FALSE;

ELSIF v\_municipio.id\_provincia = '12' AND v\_municipio.num\_habitantes >= 400 THEN

RETURN TRUE;

ELSIF v\_municipio.id\_provincia = '46' AND v\_municipio.num\_habitantes >= 650 THEN

RETURN TRUE;

ELSIF v\_municipio.id\_provincia = '03' AND v\_municipio.num\_habitantes > 500 THEN

RETURN TRUE;

END IF;

RETURN FALSE;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('NO EXISTE EL MUNICIPIO ' || p\_cod\_municipio);

RETURN FALSE;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR NO TRATADO');

RETURN FALSE;

END f\_Municipios\_Validos;

END PCK\_EJERCICIO\_PRUEBA;

/