# ÍNDICE

[ÍNDICE 1](#_Toc159875961)

[Pasos solución práctica 2](#_Toc159875962)

[Creación de tablas 3](#_Toc159875963)

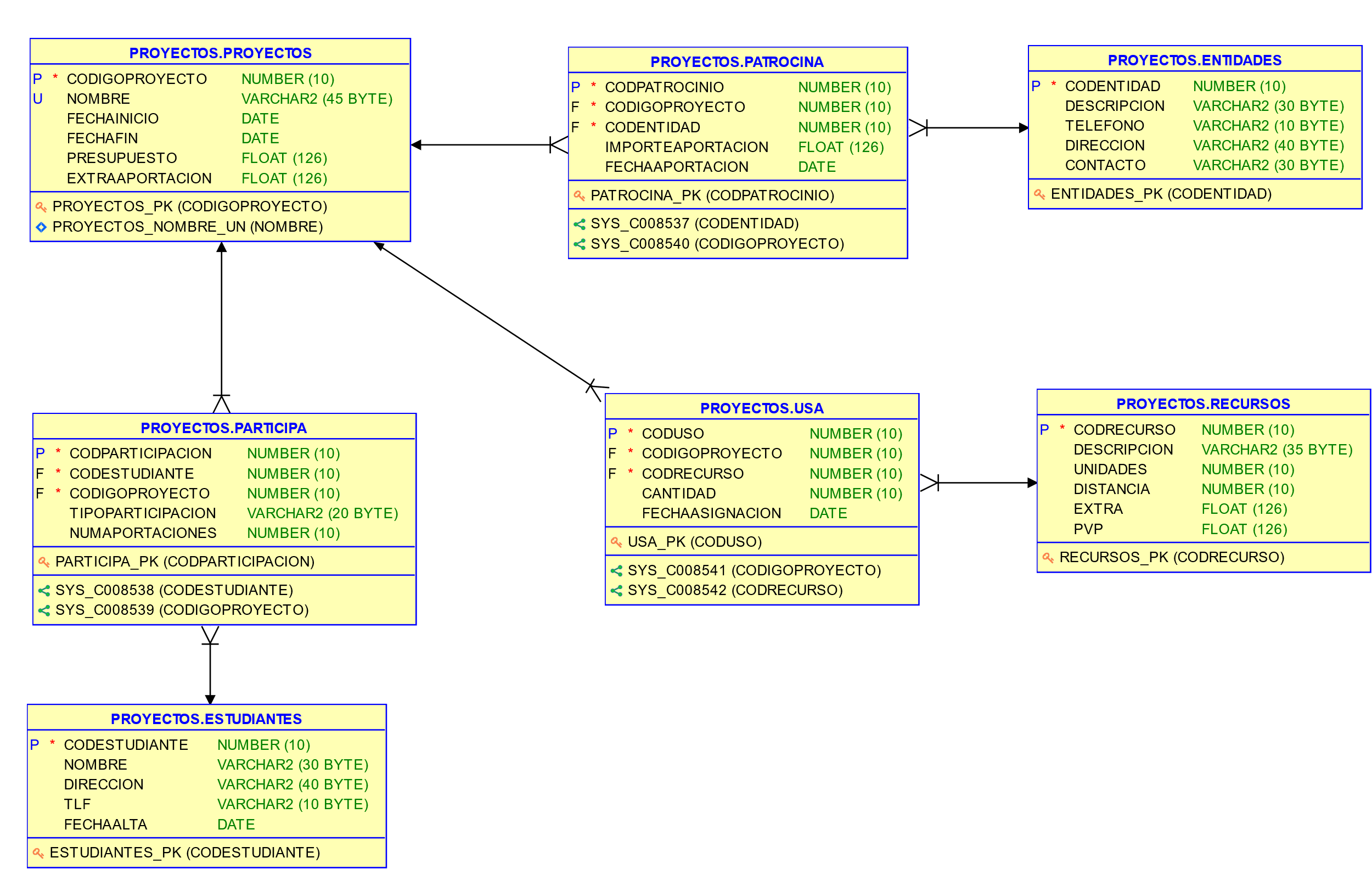
[1. Creación de tipos y tablas que serán referencia de otras tablas. 3](#_Toc159875964)

[2. Creación de tipos y definición de las tablas anidadas. 4](#_Toc159875965)

[3. Creación de la tabla que contiene las tablas anidadas 5](#_Toc159875966)

[Llenado de tablas 6](#_Toc159875967)

# Pasos solución práctica



Los requerimientos son:

1. Las Tablas PATROCINA, USA y PARTICIPA, desaparecerán en el nuevo modelo y se convertirán en tablas anidadas dentro de la tabla PROYECTOS.
2. La nueva tabla PROYECTOS tendrá los mismos atributos, y se añadirán las tablas anidadas. En estas tablas anidadas no será necesario poner el código de proyecto, pues irán dentro del proyecto.
3. Las claves ajenas de los tipos de las tablas anidadas se convertirán a referencias objetos de las tablas correspondientes.
4. El modelo nuevo creado contendrá 4 tablas nuevas de proyectos, entidades, recursos, y estudiantes.

Sabemos por el punto 2 que de las tablas PATROCINA, USA Y PARTICIPA **NO** será necesario crear referencias a la tabla proyecto cuando se definan los tipos.

Por el punto 3, sabemos que es necesario crear primero las tablas que serán referencias de las tablas anidadas.

## Creación de tablas

1. Creación de tipos y tablas que serán referencia de otras tablas.

Las tablas que se convertirán en referencia de otras tablas serán: ENTIDADES, RECURSOS y ESTUDIANTES. No evaluamos las tablas que conforme el enunciado serán tablas anidadas.

Para su creación podemos aprovechar el código del script SQL, y reutilizarlo adaptándolo a las necesidades.

**Los tipos no llevan restricciones en su definición (primary key, foreing key, unique…), si son necesarias se aplicarán sobre las tablas que usan estos tipos.**

--Borrado de tipos con dependencias de tipo o tabla

**DROP** **TYPE** fila\_entidad FORCE**;**

**DROP** **TYPE** fila\_recurso FORCE**;**

**DROP** **TYPE** fila\_estudiante FORCE**;**

--Borramos tablas si existen

**DROP** **TABLE** tbl\_entidades **cascade** **constraint;**

**DROP** **TABLE** tbl\_recursos **cascade** **constraint;**

**DROP** **TABLE** tbl\_estudiantes **cascade** **constraint;/**

--Creamos los tipos de las tablas

**CREATE** **OR** **REPLACE** **TYPE** fila\_entidad **AS** **OBJECT** **(**

codentidad NUMBER**(**10**,** 0**),**

descripcion VARCHAR2**(**30**),**

telefono VARCHAR2**(**10**),**

direccion VARCHAR2**(**40**),**

contacto VARCHAR2**(**30**)**

**);/**

**CREATE** **OR** **REPLACE** **TYPE** fila\_recurso **AS** **OBJECT** **(**

codrecurso NUMBER**(**10**,** 0**),**

descripcion VARCHAR2**(**35**),**

unidades NUMBER**(**10**,** 0**),**

distancia NUMBER**(**10**,** 0**),**

extra FLOAT**,**

pvp FLOAT

**);/**

**CREATE** **OR** **REPLACE** **TYPE** fila\_estudiante **AS** **OBJECT** **(**

codestudiante NUMBER**(**10**,** 0**),**

nombre VARCHAR2**(**30**),**

direccion VARCHAR2**(**40**),**

tlf VARCHAR2**(**10**),**

fechaalta DATE

**);/**

--Creamos las tablas

**CREATE** **TABLE** tbl\_entidades **OF** fila\_entidad **(**

codentidad **PRIMARY** **KEY**

**);/**

**CREATE** **TABLE** tbl\_recursos **OF** fila\_recurso **(**

codrecurso **PRIMARY** **KEY**

**);/**

**CREATE** **TABLE** tbl\_estudiantes **OF** fila\_estudiante **(**

codestudiante **PRIMARY** **KEY**

**);/**

1. Creación de tipos y definición de las tablas anidadas.

Tal y como menciona el enunciado las tablas del modelo relacional: PATROCINA, USA y PARTICIPA serán tablas anidadas de la tabla PROYECTOS. En este contexto, la clave foránea CODIGOPROYECTO no se evaluará ni se definirá en los tipos de las tablas anidadas.

Por lo tanto, solo serán evaluadas las claves foráneas que relacionan las tablas: ENTIDADES, RECURSOS y ESTUDIANTES.

Al definir los tipos de las tablas anidadas, las **claves foráneas** de las tablas del modelo relacional **se declaran como referencias a los tipos** definidos de las tablas relacionadas.

--Borrado de tipos con dependencias de tipo o tabla

**DROP** **TYPE** fila\_patrocina FORCE**;**

**DROP** **TYPE** fila\_usa FORCE**;**

**DROP** **TYPE** fila\_participa FORCE**;**

**DROP** **TYPE** tbl\_patrocina FORCE**;**

**DROP** **TYPE** tbl\_usa FORCE**;**

**DROP** **TYPE** tbl\_participa FORCE**;**

**/**

--Creamos los tipos de las tablas

**CREATE** **OR** **REPLACE** **TYPE** fila\_patrocina **AS** **OBJECT** **(**

codpatrocinio NUMBER**(**10**,** 0**),**

entidad **REF** fila\_entidad**,** --análogo a clave foránea

importeaportacion FLOAT**,**

fechaaportacion DATE

**);**

**/**

**CREATE** **OR** **REPLACE** **TYPE** fila\_usa **AS** **OBJECT** **(**

coduso NUMBER**(**10**,** 0**),**

recurso **REF** fila\_recurso**,** --análogo a clave foránea

cantidad NUMBER**(**10**,** 0**),**

fechaasignacion DATE

**);**

**/**

**CREATE** **OR** **REPLACE** **TYPE** fila\_participa **AS** **OBJECT** **(**

codparticipacion NUMBER**(**10**,** 0**),**

estudiante **REF** fila\_estudiante**,** --análogo a clave foránea

tipoparticipacion VARCHAR2**(**20**),**

numaportaciones NUMBER**(**10**,** 0**)**

**);**

**/**

--Creamos las tablas anidadas

**CREATE** **OR** **REPLACE** **TYPE** tbl\_ani\_patrocina **AS** **TABLE** **OF** fila\_patrocina**;**

**/**

**CREATE** **OR** **REPLACE** **TYPE** tbl\_ani\_usa **AS** **TABLE** **OF** fila\_usa**;**

**/**

**CREATE** **OR** **REPLACE** **TYPE** tbl\_ani\_participa **AS** **TABLE** **OF** fila\_participa**;**

**/**

**Los tipos, aunque sean definiciones de tabla, no tienen restricciones asociadas.**

1. Creación de la tabla que contiene las tablas anidadas

Por último, definimos el tipo aplicable a la clase PROYECTOS, y tendrán tablas anidadas.

--Borrado de tipos con dependencias de tipo o tabla

**DROP** **TYPE** fila\_proyecto FORCE**;**

**DROP** **TABLE** tbl\_proyectos **cascade** **constraint;**

--Creamos los tipos de las tablas

**CREATE** **OR** **REPLACE** **TYPE** fila\_proyecto **AS** **OBJECT** **(**

codigoproyecto NUMBER**(**10**,** 0**),**

nombre VARCHAR2**(**45**),**

fechainicio DATE**,**

fechafin DATE**,**

presupuesto FLOAT**,**

extraaportacion FLOAT**,**

patrocina tbl\_ani\_patrocina**,**

usa tbl\_ani\_usa**,**

participa tbl\_ani\_participa

**);**

**/**

**CREATE** **TABLE** tbl\_proyectos **OF** fila\_proyecto **(**

codigoproyecto **PRIMARY** **KEY,**

nombre **UNIQUE**

**)**

NESTED **TABLE** patrocina STORE **AS** tbl\_ani\_patrocina

NESTED **TABLE** usa STORE **AS** tbl\_ani\_usa

NESTED **TABLE** participa STORE **AS** tbl\_ani\_participa**;**

**/**

**NOTA: Al crear las tablas, no sería necesario especificar las restricciones de clave externa en las tablas, ya que las relaciones estarían definidas a nivel de objeto.**

## Llenado de tablas

1. Poblar tablas que son referencias de otras

Estas tablas son: ENTIDADES, RECURSOS y ESTUDIANTES.

Las sentencias para añadir el contenido de las tablas relacionales son las más sencillas al no contener ni tablas anidadas ni referencias a otros tipos de objetos.

Para evitar duplicados, primero las vaciamos y después le añadimos el contenido.

--VACIADO Y LLENADO DE TABLA ENTIDADES

**TRUNCATE** **TABLE** tbl\_entidades **DROP** STORAGE**;**

**INSERT** **INTO** tbl\_entidades **SELECT** **\*** **FROM** entidades**;**

**/**

--VACIADO Y LLENADO DE LA TABLA RECURSOS

**TRUNCATE** **TABLE** tbl\_recursos **DROP** STORAGE**;**

**INSERT** **INTO** tbl\_recursos **SELECT** **\*** **FROM** recursos**;**

**/**

--VACIADO Y LLENADO DE LA TABLA ESTUDIANTES

**TRUNCATE** **TABLE** tbl\_estudiantes **DROP** STORAGE**;**

**INSERT** **INTO** tbl\_estudiantes **SELECT** **\*** **FROM** estudiantes**;**

**/**

1. Poblar tabla que contiene tablas anidadas

Esta tabla es la tabla: PROYECTOS. Esta sentencia también es sencilla ya que vamos **a asignarle tablas anidadas vacías que posteriormente reemplazaremos**.

--VACIADO Y LLENADO DE LA TABLA PROYECTOS CON TABLAS ANIDADAS VACIAS

**TRUNCATE** **TABLE** tbl\_proyectos **DROP** STORAGE**;**

**INSERT** **INTO** tbl\_proyectos

**SELECT** p**.**codigoproyecto**,** p**.**nombre**,** p**.**fechainicio**,** p**.**fechafin**,** p**.**presupuesto**,** p**.**extraaportacion**,**

tbl\_ani\_patrocina**(),** tbl\_ani\_usa**(),** tbl\_ani\_participa**()**

**FROM** proyectos p**;**

**/**

--SI TUVIERA REFERENCIAS EN ALGUN CAMPO, PARA AÑADIR VALOR USAMOS REF(CAMPO)

--Y EN EL FROM HAY QUE PONER LA TABLA DEL CAMPO. SELECT...REF(TABLA.CAMPO) FROM ...,TABLA WHERE ...=TABLA.ID;

1. Poblar tablas anidadas

En este caso, podemos abordar el problema con varias metodologías. La primera forma sería mediante un único bloque PL/SQL y definir las consultas de inserción.

La segunda forma sería empleando la definición de procedimientos. No cabe duda que esta sería la forma más apropiada en un entorno profesional pero para un entorno educativo, donde premia el tiempo, es mejor usar el primer método.

Aún así, voy a realizarlo de las dos formas

1. Método bloque PL/SQL

**SET** SERVEROUTPUT **ON;**

**declare**

cursor proyectos **is** **select** **\*** **from** tbl\_proyectos**;**

numProyecto number**;**

**begin**

**for** fila **in** proyectos **loop**

numProyecto**:=** fila**.**codigoproyecto**;**

/\* BORRAMOS EL CONTENIDO PREVIO DE LAS TABLAS ANIDADAS \*/

**delete** **from** **table** **(**

**SELECT** tbl\_p**.**PATROCINA

**FROM** tbl\_proyectos tbl\_p

**WHERE** tbl\_p**.**codigoproyecto **=** numProyecto**);**

**delete** **from** **table** **(**

**SELECT** tbl\_p**.**USA

**FROM** tbl\_proyectos tbl\_p

**WHERE** tbl\_p**.**codigoproyecto **=** numProyecto**);**

**delete** **from** **table** **(**

**SELECT** tbl\_p**.**PARTICIPA

**FROM** tbl\_proyectos tbl\_p

**WHERE** tbl\_p**.**codigoproyecto **=** numProyecto**);**

/\* INSERTAMOS EL CONTENIDO DE LAS TABLAS ANIDADAS \*/

**INSERT** **INTO** **TABLE** **(SELECT** PATROCINA **FROM** TBL\_PROYECTOS tbl\_p

**WHERE** tbl\_p**.**codigoproyecto **=** numProyecto**)**

**(SELECT** pa**.**codpatrocinio**,** **REF(**e**),** pa**.**importeaportacion**,** pa**.**fechaaportacion

**FROM** PATROCINA pa**,** tbl\_entidades e

**WHERE** pa**.**codentidad **=** e**.**codentidad

**AND** pa**.**codigoproyecto **=** numProyecto**);**

**INSERT** **INTO** **TABLE** **(SELECT** USA **FROM** TBL\_PROYECTOS tbl\_p

**WHERE** tbl\_p**.**codigoproyecto **=** numProyecto**)**

**(SELECT** usa**.**coduso**,** **REF(**r**),** usa**.**cantidad**,** usa**.**fechaasignacion

**FROM** USA usa**,** tbl\_recursos r

**WHERE** usa**.**codrecurso **=** r**.**codrecurso

**AND** usa**.**codigoproyecto **=** numProyecto**);**

**INSERT** **INTO** **TABLE** **(SELECT** PARTICIPA **FROM** TBL\_PROYECTOS tbl\_p

**WHERE** tbl\_p**.**codigoproyecto **=** numProyecto**)**

**(SELECT** par**.**codparticipacion**,** **ref(**e**),** par**.**tipoparticipacion**,** par**.**numaportaciones

**FROM** PARTICIPA par**,** tbl\_ESTUDIANTES e

**WHERE** par**.**codestudiante **=** e**.**codestudiante

**AND** par**.**codigoproyecto **=** numProyecto**);**

**end** **loop;**

**end;**

**/**

1. Método con procedimientos

Primero creamos métodos de comprobación de existencia de datos para evitar duplicar inserción de filas.

/\* COMPROBAR SI EXISTE FILA EN TABLA ANIDADA PATROCINA \*/

**create** **or** **replace** **procedure** existePatrocinador **(**codpatrocinador number**,** codproyecto number**,** existe **out** boolean**)** **is**

CUENTA NUMBER**:=**0**;**

**begin**

**SELECT** **COUNT(\*)** **INTO** CUENTA

**FROM** **TABLE** **(SELECT** p**.**patrocina **FROM** tbl\_proyectos p

**WHERE** p**.**codigoproyecto **=** codproyecto**)** pa

**where** pa**.**codpatrocinio **=** codpatrocinador**;**

**if** CUENTA **>** 0 **then**

existe**:=true;**

**return;**

**end** **if;**

existe**:=false;**

**end;/**

/\* COMPROBAR SI EXISTE FILA EN TABLA ANIDADA USA \*/

**create** **or** **replace** **procedure** existeUso **(**codigouso number**,** codproyecto number**,** existe **out** boolean**)** **is**

CUENTA NUMBER**:=**0**;**

**begin**

**SELECT** **COUNT(\*)** **INTO** CUENTA

**FROM** **TABLE** **(SELECT** usa **FROM** tbl\_proyectos p

**WHERE** p**.**codigoproyecto **=** codproyecto**)** us

**where** us**.**coduso **=** codigouso**;**

**if** CUENTA **>** 0 **then**

existe**:=true;**

**return;**

**end** **if;**

existe**:=false;**

**end** existeUso**;/**

/\* COMPROBAR SI EXISTE FILA EN TABLA ANIDADA PARTICIPA \*/

**create** **or** **replace** **procedure** existeParticipa **(**codparticipa number**,** codproyecto number**,** existe **out** boolean**)** **is**

CUENTA NUMBER**:=**0**;**

**begin**

**SELECT** **COUNT(\*)** **INTO** CUENTA

**FROM** **TABLE** **(SELECT** p**.**participa **FROM** tbl\_proyectos p

**WHERE** p**.**codigoproyecto **=** codproyecto**)** par

**where** par**.**codparticipacion **=** codparticipa**;**

**if** CUENTA **>** 0 **then**

existe**:=true;**

**return;**

**end** **if;**

existe**:=false;**

**end;/**

Ahora definimos los procedimientos para poblar las tablas anidadas.

/\* POBLAR TABLA ANIDADA PATROCINA \*/

**CREATE** **OR** **REPLACE** **PROCEDURE** POBLAR\_TABLA\_ANIDADA\_PATROCINA **(**CODPROYECTO NUMBER**)** **AS**

--Creacion cursor: asignamos al cursor los datos de la tabla RELACIONAL

CURSOR c1 **IS**

**SELECT** pa**.**codpatrocinio**,** pa**.**codentidad**,** pa**.**importeaportacion**,**

pa**.**fechaaportacion

**FROM** PATROCINA pa

**WHERE** PA**.**CODIGOPROYECTO **=** CODPROYECTO**;**

fila c1**%ROWTYPE;**

entidad\_ref **REF** fila\_entidad**;**

existe boolean**:=false;**

filanueva fila\_patrocina**;**

**BEGIN**

-- Abrimos el cursor

**OPEN** c1**;**

**LOOP**

-- Recuperamos una fila del cursor

**FETCH** c1 **INTO** fila**;**

--Control de salida del bucle

**EXIT** **WHEN** c1**%**NOTFOUND**;**

--Evaluamos si ya existe el registro en la tabla anidada

existePatrocinador**(**fila**.**codpatrocinio**,** codproyecto**,** existe**);**

--Si no existe creamos el registro y lo asignamos a la tabla anidada

de tipo OBJETO

**if** **not** existe **then**

-- Obtenemos la referencia de la entidad

**SELECT** **REF(**e**)** **INTO** entidad\_ref

**FROM** TBL\_ENTIDADES e

**WHERE** e**.**codentidad **=** fila**.**codentidad**;**

filanueva **:=** fila\_patrocina**(**fila**.**codpatrocinio**,** entidad\_ref**,**

fila**.**importeaportacion**,** fila**.**fechaaportacion**);**

-- Insertamos filas

**INSERT** **INTO** **TABLE** **(SELECT** t**.**PATROCINA **FROM** TBL\_PROYECTOS t

**WHERE** t**.**codigoproyecto **=** CODPROYECTO**)**

**VALUES** **(**filanueva**);**

**else**

dbms\_output**.**put\_line**(**'Ya existe la fila con número de patrocinio: ' **||**

fila**.**codpatrocinio **||** ', y no se insertará.'**);**

**end** **if;**

existe**:=false;**

**END** **LOOP;**

**CLOSE** c1**;** -- Cerramos el cursor

**END;**

**/**

/\* POBLAR TABLA ANIDADA USA \*/

**CREATE** **OR** **REPLACE** **PROCEDURE** POBLAR\_TABLA\_ANIDADA\_USA **(**CODPROYECTO NUMBER**)** **AS**

--Creacion cursor: asignamos al cursor los datos de la tabla RELACIONAL

CURSOR c1 **IS**

**SELECT** us**.**coduso**,** us**.**codrecurso**,** us**.**cantidad**,** us**.**fechaasignacion

**FROM** USA us

**WHERE** us**.**CODIGOPROYECTO **=** CODPROYECTO**;**

fila c1**%ROWTYPE;**

recurso\_ref **REF** fila\_recurso**;**

existe boolean**:=false;**

filanueva fila\_usa**;**

**BEGIN**

-- Abrimos el cursor

**OPEN** c1**;**

**LOOP**

-- Recuperamos una fila del cursor

**FETCH** c1 **INTO** fila**;**

--Control de salida del bucle

**EXIT** **WHEN** c1**%**NOTFOUND**;**

--Evaluamos si ya existe el registro en la tabla anidada

existeUso**(**fila**.**coduso**,** codproyecto**,** existe**);**

--Si no existe creamos el registro y lo asignamos a la tabla anidada

de tipo OBJETO

**if** **not** existe **then**

-- Obtenemos la referencia de la entidad

**SELECT** **REF(**r**)** **INTO** recurso\_ref

**FROM** tbl\_recursos r

**WHERE** r**.**codrecurso **=** fila**.**codrecurso**;**

filanueva **:=** fila\_usa**(**fila**.**coduso**,** recurso\_ref**,** fila**.**cantidad**,**

fila**.**fechaasignacion**);**

-- Insertamos filas

**INSERT** **INTO** **TABLE** **(SELECT** t**.**USA **FROM** TBL\_PROYECTOS t

**WHERE** t**.**codigoproyecto **=** CODPROYECTO**)**

**VALUES** **(**filanueva**);**

**else**

dbms\_output**.**put\_line**(**'Ya existe la fila con número de usa: ' **||**

fila**.**coduso **||** ', y no se insertará.'**);**

**end** **if;**

existe**:=false;**

**END** **LOOP;**

**CLOSE** c1**;** -- Cerramos el cursor

**END;**

**/**

/\* POBLAR TABLA ANIDADA PARTICIPA \*/

**CREATE** **OR** **REPLACE** **PROCEDURE** POBLAR\_TABLA\_ANIDADA\_PARTICIPA **(**CODPROYECTO NUMBER**)** **AS**

--Creacion cursor: asignamos al cursor los datos de la tabla RELACIONAL

CURSOR c1 **IS**

**SELECT** par**.**codparticipacion**,** par**.**codestudiante**,** par**.**tipoparticipacion**,**

par**.**numaportaciones

**FROM** PARTICIPA PAR

**WHERE** PAR**.**CODIGOPROYECTO **=** CODPROYECTO**;**

fila c1**%ROWTYPE;**

estudiante\_ref **REF** fila\_estudiante**;**

existe boolean**:=false;**

filanueva fila\_participa**;**

**BEGIN**

-- Abrimos el cursor

**OPEN** c1**;**

**LOOP**

-- Recuperamos una fila del cursor

**FETCH** c1 **INTO** fila**;**

--Control de salida del bucle

**EXIT** **WHEN** c1**%**NOTFOUND**;**

--Evaluamos si ya existe el registro en la tabla anidada

existeParticipa**(**fila**.**codparticipacion**,** codproyecto**,** existe**);**

--Si no existe creamos el registro y lo asignamos a la tabla anidada

de tipo OBJETO

**if** **not** existe **then**

-- Obtenemos la referencia de la entidad

**SELECT** **REF(**e**)** **INTO** estudiante\_ref

**FROM** TBL\_ESTUDIANTES e

**WHERE** e**.**codestudiante **=** fila**.**codestudiante**;**

filanueva **:=** fila\_participa**(**fila**.**codparticipacion**,** estudiante\_ref**,**

fila**.**tipoparticipacion**,** fila**.**numaportaciones**);**

-- Insertamos filas

**INSERT** **INTO** **TABLE** **(SELECT** t**.**PARTICIPA **FROM** TBL\_PROYECTOS t

**WHERE** t**.**codigoproyecto **=** CODPROYECTO**)**

**VALUES** **(**filanueva**);**

**else**

dbms\_output**.**put\_line**(**'Ya existe la fila con número de participacion: ' **||**

fila**.**codparticipacion **||** ', y no se insertará.'**);**

**end** **if;**

existe**:=false;**

**END** **LOOP;**

**CLOSE** c1**;** -- Cerramos el cursor

**END;**

**/**

Como se aprecia la extensión de código usando procedimientos es significativa y traducido en tiempo puede marcar un antes y un después.