**TRABAJO FINAL 2ª Ev:** Acceso a Datos – PL/SQL

Grupo: **2ºSI** Curso: 23/24

Fecha: 26/02/24

**I. Instrucciones generales**

*Lee con atención todo el enunciado*, de la primera a la última cláusula.

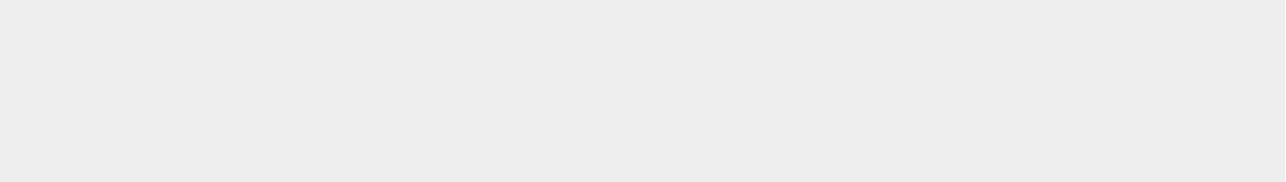
Es probable que en una primera o segunda lecturas no se entienda todo al detalle. Empieza trabajando aquellos puntos que entiendas mejor, y poco a poco irás entendiendo mejor los restantes.

Las secciones I y II son informativas. La sección III contiene la parte que debe implementarse de forma obligatoria. La sección IV es opcional.

**II. Tablas**

Nuestro escenario será la BD de una biblioteca

01. La tabla *obra* contiene la información de las obras literarias.



CREATE TABLE obra (

id CHAR(5),

titulo VARCHAR(100),

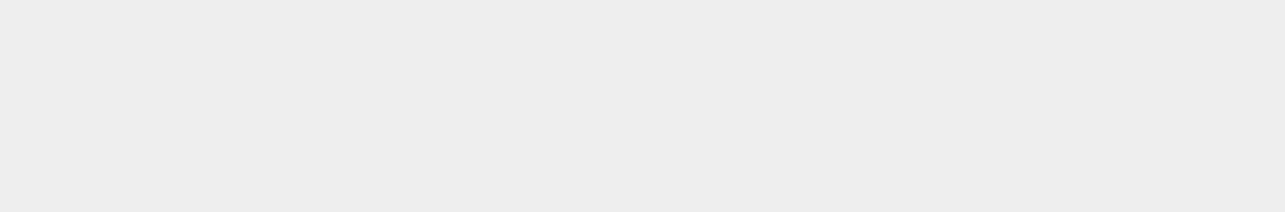
anyo INTEGER,

CONSTRAINT PK\_obra PRIMARY KEY (id),

CONSTRAINT NN\_titulo CHECK (titulo IS NOT NULL)

);

02. La tabla *autor* contiene la información de los autores de las obras.



CREATE TABLE autor (

id CHAR(4),

nombre VARCHAR(30),

apellidos VARCHAR(60),

nacimiento DATE,

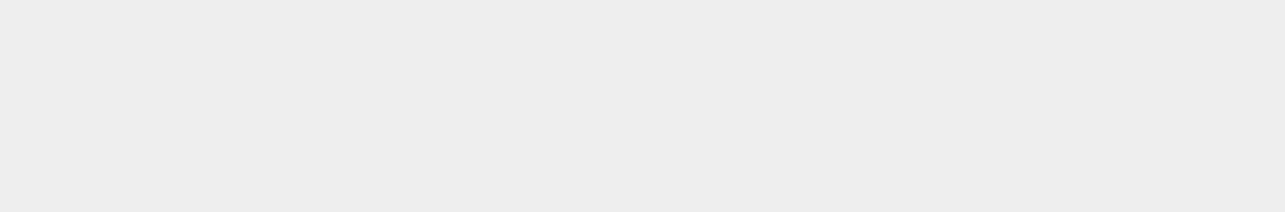
CONSTRAINT PK\_autor PRIMARY KEY (id),

CONSTRAINT NN\_nombre CHECK (nombre IS NOT NULL),

CONSTRAINT NN\_apellidos CHECK (apellidos IS NOT NULL)

);

03. La tabla autor\_obra contiene la relación de autores y obras.



CREATE TABLE autor\_obra (

id\_autor CHAR(4),

id\_obra CHAR(5),

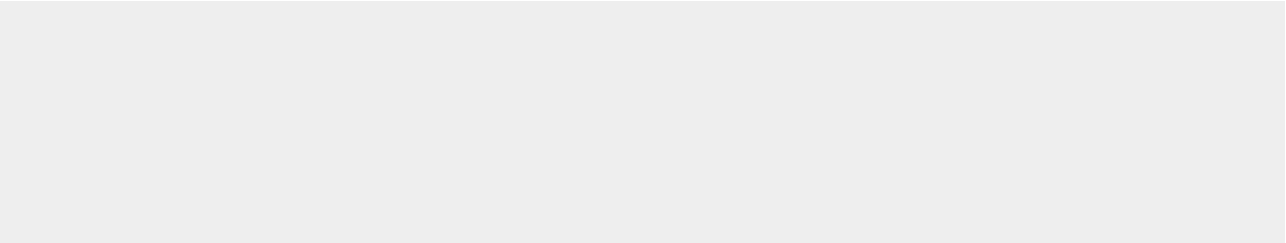
CONSTRAINT PK\_autor\_obra PRIMARY KEY (id\_autor, id\_obra),

CONSTRAINT FK\_autor\_obra\_id\_autor FOREIGN KEY (id\_autor) REFERENCES autor(id),

CONSTRAINT FK\_autor\_obra\_id\_obra FOREIGN KEY (id\_obra) REFERENCES obra(id)

);

04. La tabla *edicion* contiene la información de las ediciones publicadas de las obras.



CREATE TABLE edicion (

id CHAR(6),

id\_obra CHAR(5),

isbn VARCHAR(20),

anyo INTEGER,

CONSTRAINT PK\_edicion PRIMARY KEY (id),

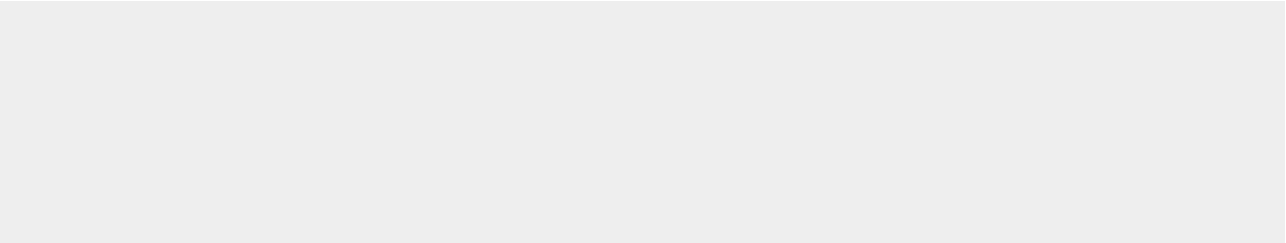
CONSTRAINT NN\_id\_obra CHECK (id\_obra IS NOT NULL),

CONSTRAINT NN\_isbn CHECK (isbn IS NOT NULL),

CONSTRAINT FK\_edicion FOREIGN KEY (id\_obra) REFERENCES obra(id)

);

05. La tabla *ejemplar* contiene la información de los ejemplares adquiridos por la biblioteca.



CREATE TABLE ejemplar (

id\_edicion CHAR(6),

numero INTEGER,

alta DATE,

baja DATE,

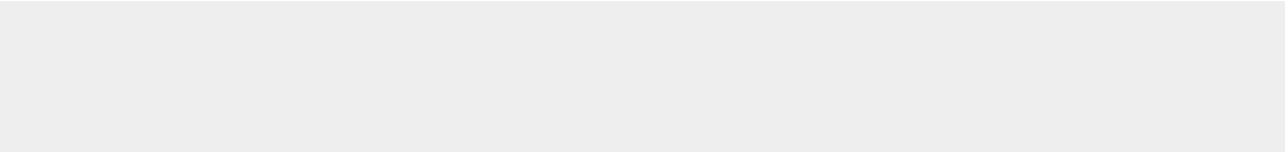
CONSTRAINT PK\_ejemplar PRIMARY KEY (id\_edicion, numero),

CONSTRAINT FK\_ejemplar FOREIGN KEY (id\_edicion) REFERENCES edicion(id),

CONSTRAINT NN\_alta CHECK (alta IS NOT NULL)

);

06. Los identificadores serán alfanuméricos y generados de forma aleatoria. Por supuesto habrá que tener en cuenta antes de realizar la inserción que la PK no esté duplicada, caso de estarlo terminaremos de forma controlada el intento de inserción monitorizando por pantalla el error.



DECLARE

id5 CHAR(5);

BEGIN

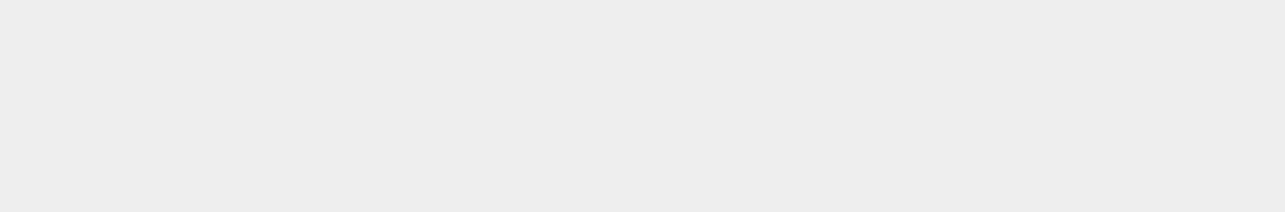
id5:= dbms\_random.string('X', 5); -- insertar una obra

INSERT INTO obra (id, titulo) VALUES (id5, 'Cinco semanas en globo');

END;

07. Los ejemplares *de cada edición* serán numerados de forma consecutiva.

**PREGUNTA:** ¿De donde vienen estos datos? ¿Hay que insertarlos a mano?



SELECT \* FROM ejemplar WHERE id\_edicion = 'FSDPC7';

|  |  |  |  |
| --- | --- | --- | --- |
| id\_edicion numero | | alta | baja |
| ---------- ------ | | ---------- | ---------- |
| FSDPC7 | 1 | 2009/07/27 | 2013/08/05 |
| FSDPC7 | 2 | 2009/07/27 | (NULL) |
| FSDPC7 | 3 | 2013/08/29 | (NULL) |

**III. Operativa requerida**

08. El alta de obras se implementará mediante una función *alta\_obra*.



-------------------------------------------------------------------------------

* efectos: alta de obra con título y año
* retorno: id asignado, ‘-1’ error

-------------------------------------------------------------------------------

FUNCTION alta\_obra(p\_titulo VARCHAR, p\_anyo INTEGER DEFAULT NULL)

RETURN VARCHAR IS

* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

09. El borrado de obras se implementará mediante una función *borrado\_obra*.



-------------------------------------------------------------------------------

* efectos: borrado de obra por id
* retorno: 1 si borrado efectuado, 0 si no existe id, -1 error

-------------------------------------------------------------------------------

FUNCTION borrado\_obra(p\_id VARCHAR) RETURN INTEGER IS

* escriba aquí las declaraciones

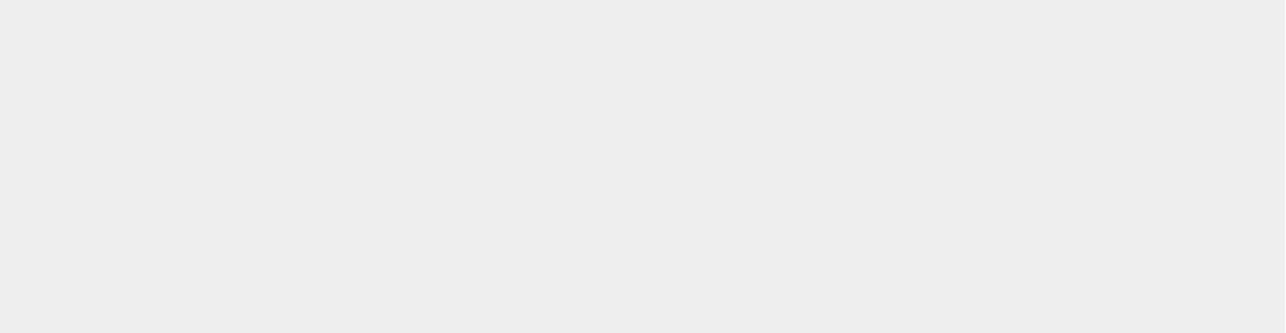
BEGIN

* escriba aquí el código

END;

09.1 Auditoría del borrado de obras: Cada vez que se ejecute el borrado de una obra y para no perder datos, lo que querremos es que la obra borrada y todos los datos asociados a ella queden guardados en una/s tabla/s de histórico ocultas al usuario estándar, de modo que en cualquier momento se pueda auditar el borrado de obras recuperando todos sus datos y la fecha/hora en que fue borrada la obra.

**PREGUNTA: ¿Hay qué hacer ambos triggers o vale con solo hacer uno?**



-------------------------------------------------------------------------------

* efectos: Auditoría del borrado de obra por id
* retorno: No aplicable (TRIGGER)

-------------------------------------------------------------------------------

TRIGGER auditar\_borrado\_obra BEFORE/AFTER DELETE ON . . .

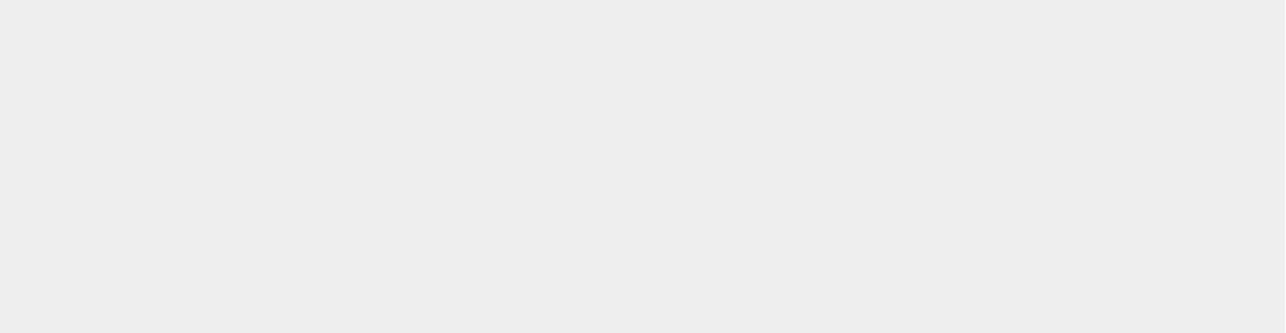
* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

10. El alta de autores se implementará mediante una función *alta\_autor*.



-------------------------------------------------------------------------------

* efectos: alta de autor con nombre, apellidos y nacimiento
* retorno: id asignado, ‘-1’ error

-------------------------------------------------------------------------------

FUNCTION alta\_autor(p\_nombre VARCHAR, p\_apellidos VARCHAR,

p\_nacimiento DATE DEFAULT NULL)

RETURN VARCHAR IS

* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

11. El borrado de autores se implementará mediante una función *borrado\_autor*.



-------------------------------------------------------------------------------

* efectos: borrado de autor por id
* retorno: 1 si borrado efectuado, 0 si no existe id, -1 error

-------------------------------------------------------------------------------

FUNCTION borrado\_autor(p\_id VARCHAR) RETURN INTEGER IS

* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

**PREGUNTA: SI PASO UNA ID VACIA (BORRADO\_AUTOR(‘ ’) SE LANZA UNA EXCEPCION DE NO\_DATA\_FOUND. SE DEJA ASÍ O SE HACEN VERIFICACIONES PARA SABER SI LA ID QUE SE ESTA PASANDO ES UN CAMPO VACIO O NO?)**

12. La vinculación de un autor a una obra se implementará mediante una función *vincular*.



-------------------------------------------------------------------------------

* efectos: vincula un autor a una obra
* retorno: 1 (sin error) , -1 error

-------------------------------------------------------------------------------

FUNCTION vincular(p\_id\_autor VARCHAR, p\_id\_obra VARCHAR) RETURN INTEGER IS

* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

13. La desvinculación de un autor de una obra se implementará mediante una función *desvincular*.



-------------------------------------------------------------------------------

* efectos: desvincula un autor de una obra
* retorno: 1 si desvinculación efectuada, 0 si no existe vínculo, -1 error

-------------------------------------------------------------------------------

FUNCTION desvincular(p\_id\_autor VARCHAR, p\_id\_obra VARCHAR)

RETURN INTEGER IS

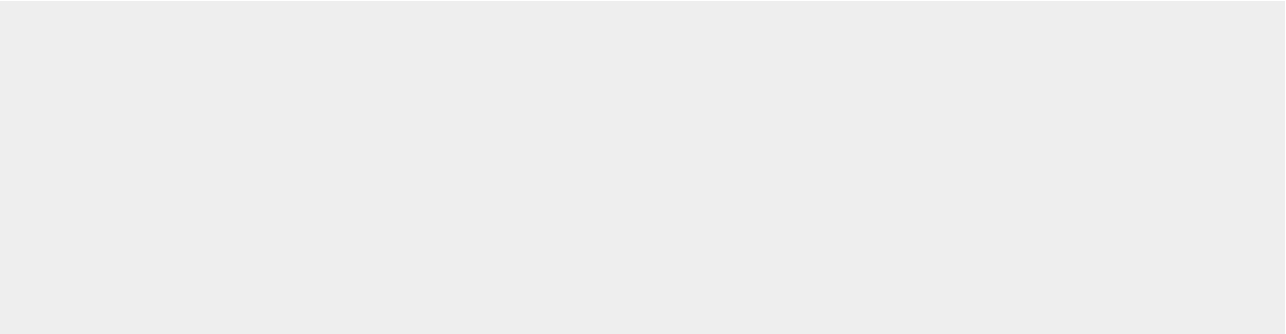
* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

14. El alta de ediciones se implementará mediante una función *alta\_edicion*.



-------------------------------------------------------------------------------

* efectos: alta de edición con obra, isbn y año
* retorno: id asignado, ‘-1’ error

-------------------------------------------------------------------------------

FUNCTION alta\_edicion(p\_id\_obra VARCHAR, p\_isbn VARCHAR,

p\_año INTEGER DEFAULT NULL)

RETURN VARCHAR IS

* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

15. El borrado de ediciones se implementará mediante una función *borrado\_edicion*.



-------------------------------------------------------------------------------

* efectos: borrado de edición por id
* retorno: 1 si borrado efectuado, 0 si no existe id

-------------------------------------------------------------------------------

FUNCTION borrado\_edicion(p\_id VARCHAR)

RETURN INTEGER IS

* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

16. El alta de ejemplares se implementará mediante una función *alta\_ejemplar*.



-------------------------------------------------------------------------------

* efectos: alta de ejemplar con id\_edicion
* retorno: número asignado, -1 si no existe id\_edicion.

-------------------------------------------------------------------------------

FUNCTION alta\_ejemplar(p\_id\_edicion VARCHAR) RETURN INTEGER IS

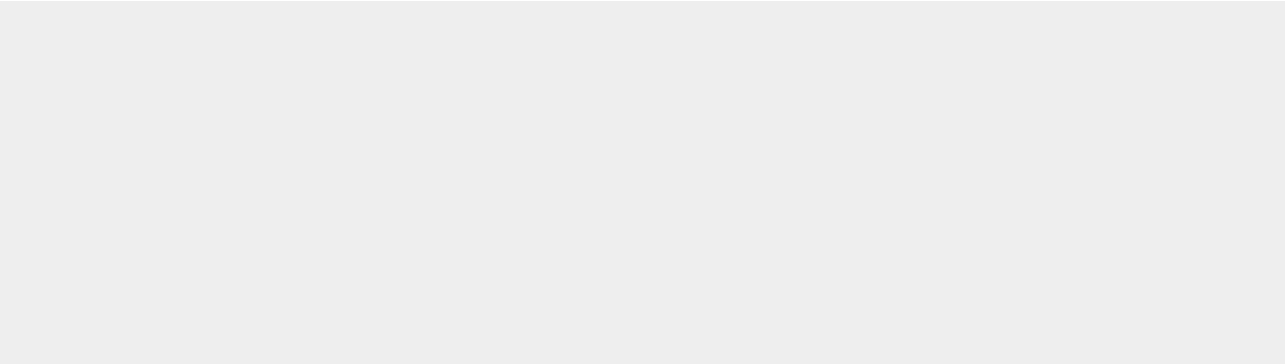
* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

1. La operación de alta de ejemplar no requiere especificar fecha de alta, porque será la fecha del sistema.
2. La operación de alta de ejemplar no requiere especificar el número, porque se calculará según los ejemplares de la misma edición ya existentes (si ya hay N, pues se dará de alta como N+1).
3. Para los ejemplares se distinguirá entre las operaciones de *borrado* y de *baja*, ya que tienen diferente significado y aplicación.
4. El borrado de ejemplares significa suprimir la fila correspondiente, y su aplicación habitual será revertir un alta errónea.
5. **Sólo se podrá borrar un ejemplar si es el último de su serie, no tiene fecha de baja y además no han pasado más de 30 días desde la fecha de alta.**
6. El borrado de ejemplares se implementará mediante una función *borrado\_ejemplar*.



-------------------------------------------------------------------------------

* efectos: borrado de ejemplar por id\_edicion y número
* retorno: 1 si borrado efectuado (cumpliendo cláusula 21),
* 0 borrado no efectuado (no existe el ejemplar indicado)
* -1 borrado no efectuado (existe pero no cumple la cláusula 21)

-------------------------------------------------------------------------------

FUNCTION borrado\_ejemplar(p\_id\_edicion VARCHAR, p\_numero INTEGER) RETURN INTEGER IS

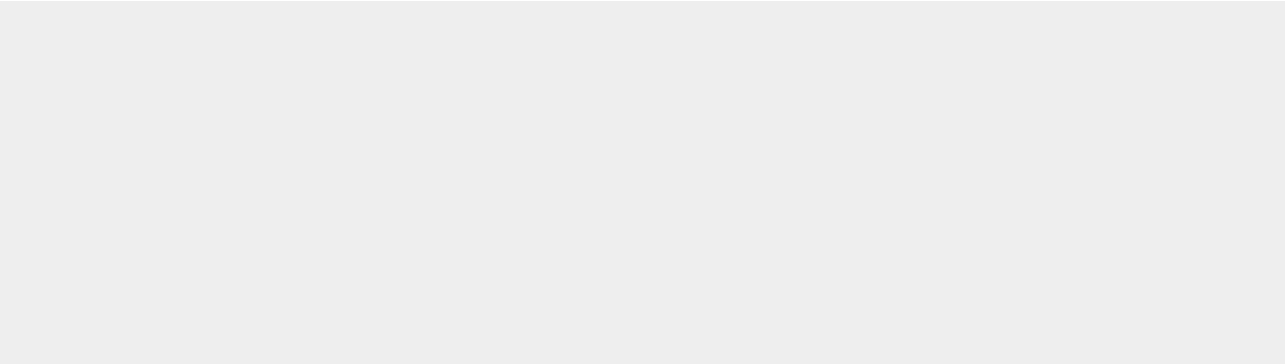
* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

1. La baja de un ejemplar consistirá en establecer la fecha de baja, y su aplicación será reflejar que éste ha sido retirado del sistema por deterioro o extravío.
2. La operación de baja no requiere especificar la fecha, porque se utilizará la del sistema.
3. Sólo se podrá dar de baja un ejemplar si no tiene fecha de baja.
4. La baja de ejemplares se implementará mediante una función *baja\_ejemplar*.



-------------------------------------------------------------------------------

* efectos: baja de ejemplar por id\_edicion y número
* retorno: 1 si baja efectuada (cumpliendo la cláusula 25)
* 0 baja no efectuada (no existe el ejemplar indicado)
* -1 baja no efectuada (existe pero no cumple la cláusula 25)

-------------------------------------------------------------------------------

FUNCTION baja\_ejemplar(p\_id\_edicion VARCHAR, p\_numero INTEGER) RETURN INTEGER IS

* escriba aquí las declaraciones

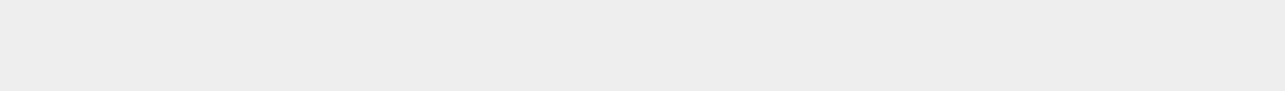
BEGIN

* escriba aquí el código

END;

**IV. Operativa opcional**

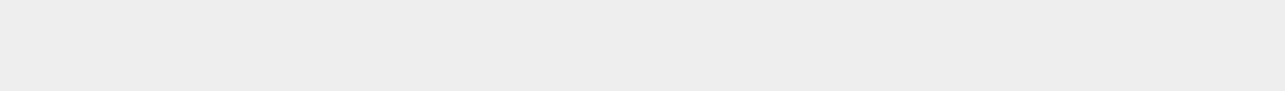
27. Definir una tabla con la información (nombre y apellidos) de los socios de la biblioteca.



CREATE TABLE socio (

-- escriba aquí la definición de la tabla

);

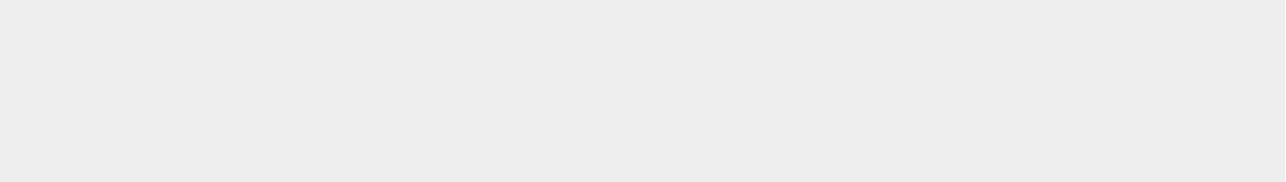
28. Definir una tabla con la información de los préstamos.

CREATE TABLE prestamo (

-- escriba aquí la definición de la tabla

);

29. Escriba una función llamada *apertura\_prestamo* para abrir un préstamo de libro.



FUNCTION apertura\_prestamo ( <parámetros-que-estime-oportunos> )

RETURN <tipo-de-retorno-que-estime-oportuno> IS

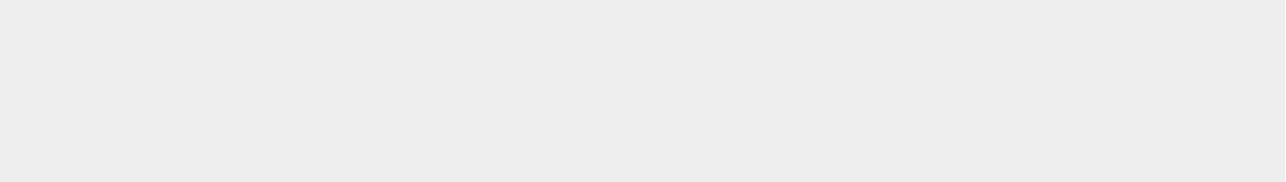
* escriba aquí las declaraciones

BEGIN

* escriba aquí el código

END;

30. Auditar el borrado de cualquier elemento de la BD tal y como realizamos en el punto 9.1 a través de TRIGGERS

30. Escriba una función llamada *cierre\_préstamo* para cerrar un préstamo de libro.

FUNCTION cierre\_prestamo ( <parámetros-que-estime-oportunos> )

RETURN <tipo-de-retorno-que-estime-oportuno> IS

* escriba aquí las declaraciones

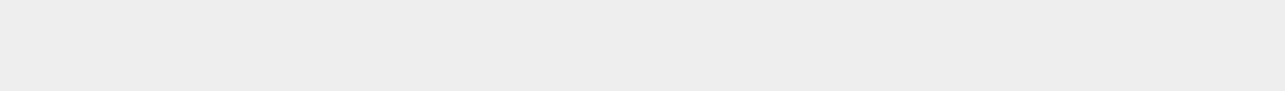
BEGIN

* escriba aquí el código

END;

**V. Formato de entrega**

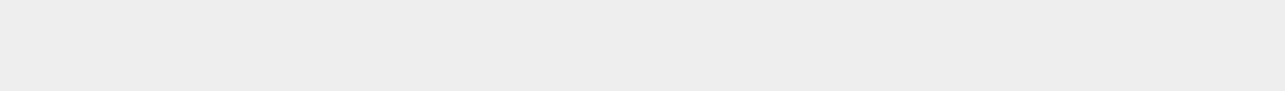
1. Deberá entregar un fichero ZIP con el formato APELLIDOS\_NOMBRE.ZIP, con el código fuente PL/SQL que haya desarrollado en dos o tres ficheros de texto que contendrán los procedimientos y funciones realizados agrupados en un Package de PLSQL y un tercer fichero que contendrá la parte opcional si se ha decidido llevarla a cabo. El cuarto fichero será la batería de pruebas que se ha realizado a nuestros procedimientos y/o funciones de los packetes (ver punto 32, criterios de evaluación)
2. El primer fichero se llamará BIBLIOTK\_PACKAGE.SQL, con el siguiente contenido.



CREATE OR REPLACE PACKAGE BiblioTK AS

... las funciones tal como deben aparecer en CREATE PACKAGE END;

29. El segundo fichero se llamará BIBLIOTK\_BODY. SQL, con el siguiente contenido.



CREATE OR REPLACE PACKAGE BODY BiblioTK AS

... las funciones tal y como deben aparecer en CREATE PACKAGE BODY END;

1. El tercer fichero se llamará BIBLIOTK\_ESQUEMA\_PARTE2. SQL y contendrá la definición CREATE TABLE de las tablas *socio* y *préstamo* de la operativa opcional, si la implementa.

**VI. Criterios de evaluación**

1. Los ficheros entregados se ejecutarán en un entorno SQL para verificar que compilan y/o ejecutan correctamente.
2. Una vez introducidos en el entorno SQL, se ejecutará una **batería** **invocaciones** de las funciones del paquete BIBLIOTK implementado para verificar si cumplen las especificaciones.
3. Las cláusulas de la sección III (operativa requerida) califican por un 70% de la nota y su implementación es obligatoria (por tanto, prioritaria).
4. Las cláusulas de la sección IV (operativa opcional) califican por un 30% de la nota y su implementación es opcional (por tanto, secundaria).

**VII. Fecha límite de entrega**

Lunes 11 de Marzo de 2024.