BASES DE DATOS 20/MAYO/2019
GRUPO 1 PROYECTO


- CHAVIRA TAPIA ANDRES URIEL
- JOYA VENEGAS JEHOSUA ALAN
- LÁZARO MARTÍNEZ ABRAHAM JOSUE
- MENDOZA TOLEDO OSCAR


En una biblioteca se necesita llevar el control del préstamo de su material a sus lectores. Todo material cuenta con un identificador, titulo, colocación, ubicación dentro de la biblioteca y autor; los autores están registrados con una clave de autor, nombre completo y nacionalidad. Se tienen dos tipos de materiales: libros y tesis; de los libros se tiene número de adquisición que es único, ISBN, tema y edición; las tesis están registradas con un identificador de tesis, carrera del tema de tesis, año de publicación, así como el director de tesis. Del director de tesis se tiene un identificador único, nombre completo y grado académico. De cada material se tiene por lo menos un ejemplar, para cada ejemplar se tiene número de ejemplar y estatus que indica si está disponible, en préstamo, no sale de la biblioteca o si esta en mantenimiento. Los lectores están registrados por un identificador, tipo de lector (alumno, profesor, investigador), nombre completo, dirección, teléfono, adeudo, fecha en la cual se dio de alta en la biblioteca y fecha de vigencia (un año a partir de la fecha de alta en la biblioteca), dependiendo del tipo de lector es el límite de materiales que puede solicitar en préstamo, días permitidos, así como número de refrendos autorizados.

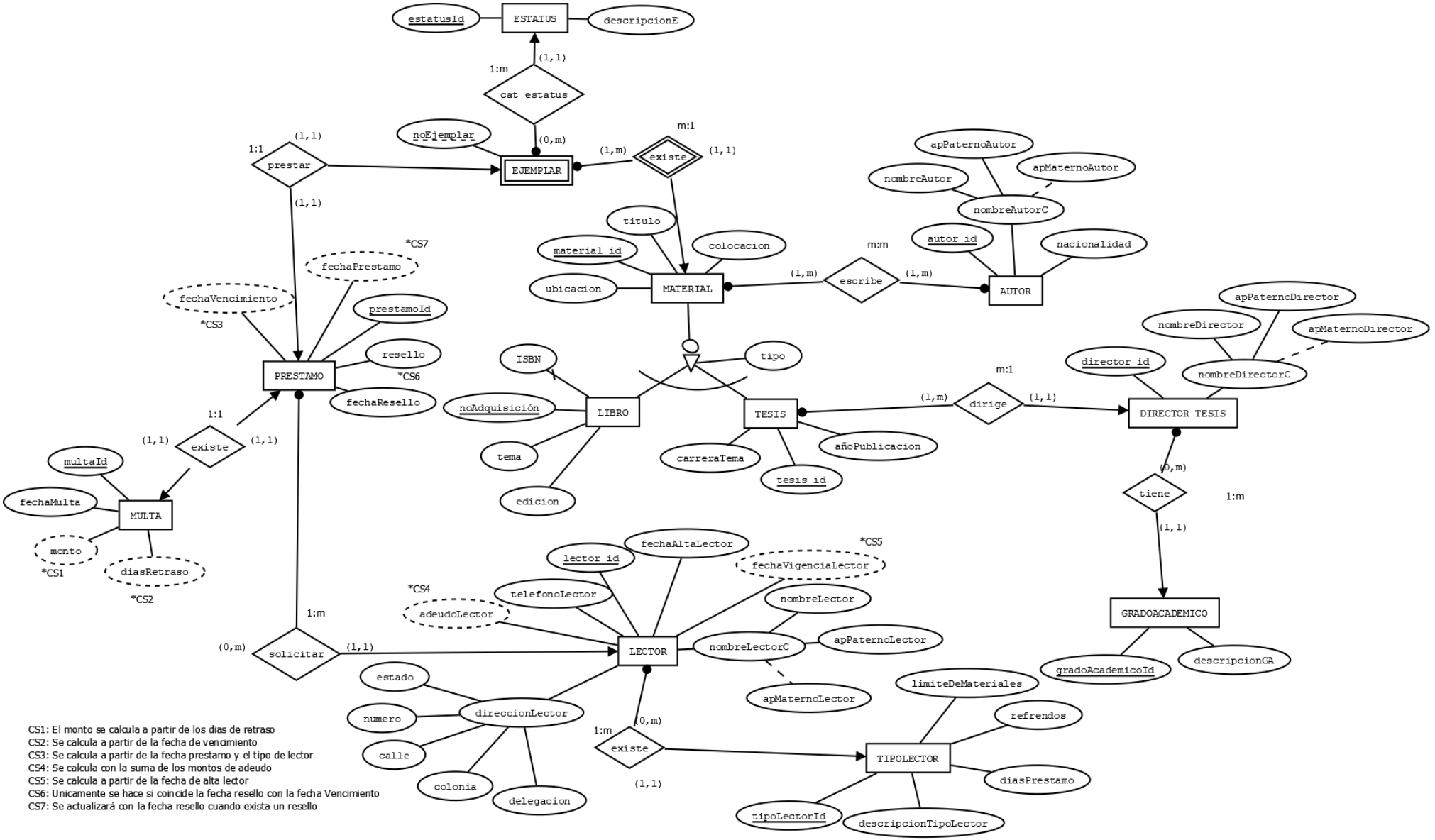
| Lector | Límite de materiales | Días de préstamo | Refrendos |
|---|---|---|---|
| Estudiante | 3 | 8 | 1 |
| Profesor | 5 | 15 | 2 |
| Investigador | 10 | 30 | 3 |

Al realizarse un préstamo es indispensable almacenar información sobre la fecha en la que se realiza, así como la fecha de vencimiento en base al tipo de lector; al realizarse una devolución es importante considerar que la fecha de devolución no exceda de la fecha de vencimiento o de lo contrario se registrará una multa (10 pesos por día de atraso) en la cual se debe de considerar el lector que lo solicito, el material en préstamo, la fecha de la multa, número de días de retraso y el monto al que se hizo acreedor, en el caso de resello se debe de considerar el número de refrendos permitidos de acuerdo al tipo de lector. Los refrendos se realizan únicamente en la fecha de vencimiento

Consideraciones

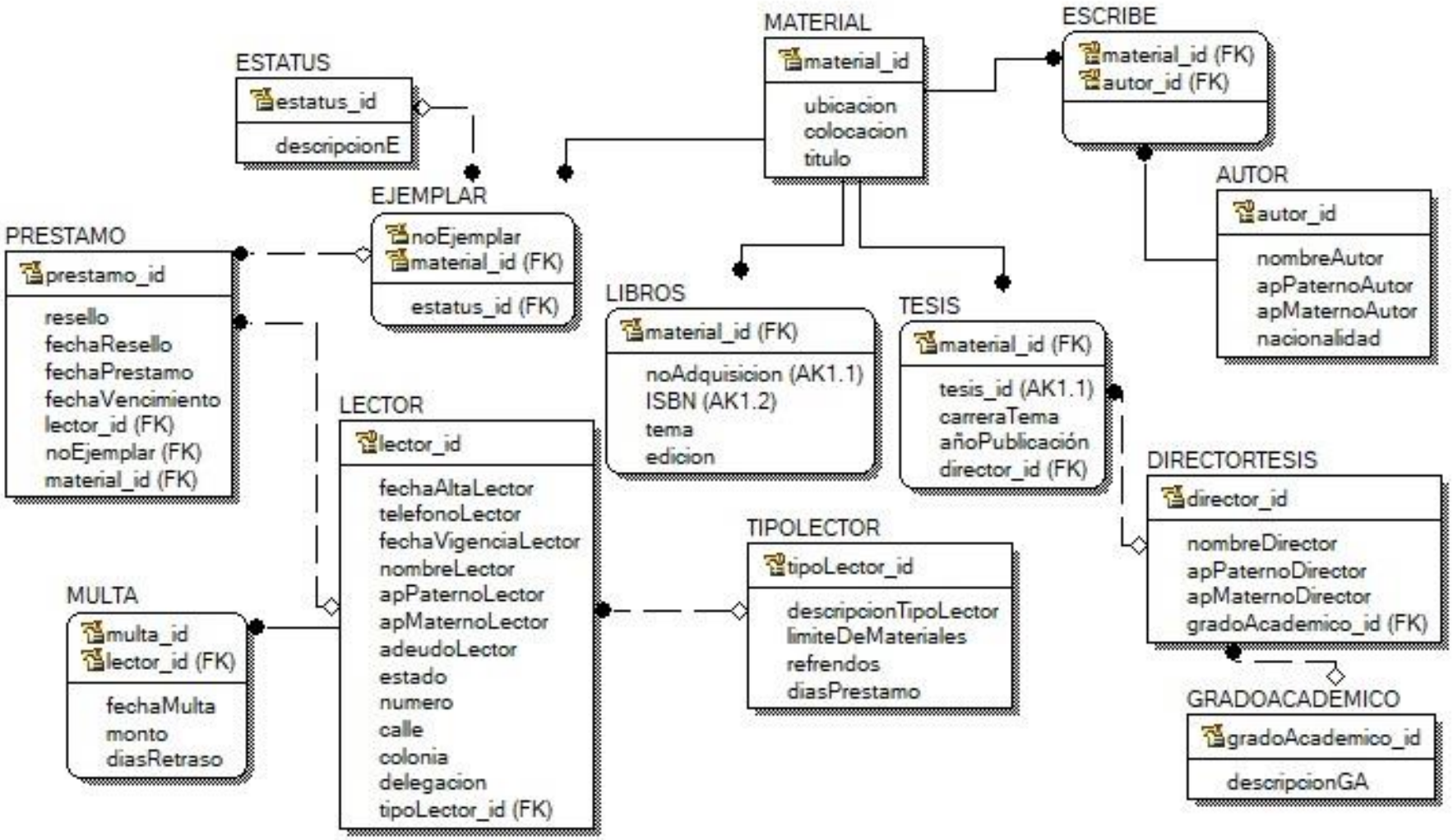
- Si un lector tiene una multa, no se le podrán prestar materiales hasta que la haya liquidado.
- Al realizarse el préstamo de un ejemplar, se deberá de modificar su estatus automáticamente.
- El resello de un material se realiza únicamente en la fecha de vencimiento del préstamo en función del tipo de lector. Al realizarse una devolución en tiempo, se eliminará el préstamo
- Al resellar el préstamo de un material, la fecha del préstamo cambiará a la fecha en la que se resella, la fecha de vencimiento se volverá a calcular dependiendo del tipo de lector y se actualizará el número de refrendo.
- Deben existir vistas que muestren los catálogos de libros, de tesis, autores con su tipo de material, estatus del material y usuarios por tipo.

# Modelo ER




CS1: El monto se calcula a partir de los dias de retraso
CS2: Se calcula a partir de la fecha de vencimiento
CS3: Se calcula a partir de la fecha prestamo y el tipo de lector
CS4: Se calcula con la suma de los montos de adeudo
CS5: Se calcula a partir de la fecha de alta lector
CS6: Unicamente se hace si coincide la fecha resello con la fecha Vencimiento
CS7: Se actualizará con la fecha resello cuando exista un resello

# Modelo Relacional



AUTOR

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integridad | Derivado |
|---|---|---|---|---|---|---|---|
| autor_id | CHAR(10) | PK | No | No | Si | PK | No |
| nombreAutor | VARCHAR2(20) | No | No | No | Si | No | No |
| apPaternoAutor | VARCHAR2(20) | No | No | No | Si | No | No |
| apMaternoAutor | VARCHAR2(20) | No | No | No | No | No | No |
| nacionalidad | VARCHAR2(15) | No | No | No | Si | No | No |

MATERIAL

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integr | Derivado |
|---|---|---|---|---|---|---|---|
| material_id | CHAR(5) | PK | No | No | Si | PK | No |
| ubicacion | VARCHAR2(20) | No | No | No | Si | No | No |
| colocacion | VARCHAR2(20) | No | No | No | Si | No | No |
| titulo | VARCHAR2(100) | No | No | No | Si | No | No |
| tipoMaterial | CHAR(1) | No | Si | No | Si | No | No |

ESCRIBE

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integr | Derivado |
|---|---|---|---|---|---|---|---|
| material_id | CHAR(5) | FK | No | No | Si | PK,Fk nulo | No |
| autor_id | CHAR(10) | FK | No | No | Si | PK, Fk nulo | No |

LIBRO

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integr | Derivado |
|---|---|---|---|---|---|---|---|
| material_id | CHAR(5) | FK | No | No | Si | PK,Fk nulo | No |
| noAdquisicion | CHAR(10) | U | No | No | Si | No | No |
| ISBN | CHAR(16) | U | No | No | Si | No | No |
| tema | VARCHAR2(50) | No | No | No | Si | No | No |
| edicion | VARCHAR2(20) | No | No | No | Si | No | No |

GRADOACADEMICO

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integr | Derivado |
|---|---|---|---|---|---|---|---|
| gradoAcademico_id | VARCHAR2(15) | PK | No | No | Si | PK | No |
| descripcionGA | VARCHAR2(20) | No | No | No | Si | No | No |

DIRECTOR TESIS

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integr | Derivado |
|---|---|---|---|---|---|---|---|
| director_id | CHAR(10) | PK | No | No | Si | PK | No |
| nombreDirector | VARCHAR2(20) | No | No | No | Si | No | No |
| apPaternoDirector | VARCHAR2(20) | No | No | No | Si | No | No |
| apMaternoDirector | VARCHAR2(20) | No | No | No | No | No | No |
| gradoAcademico_id | VARCHAR2(15) | FK | No | No | Si | Fk, nulo | No |

TESIS

| Nombre | Tipo de dato | Llave | Rest de dom | Omision | Obligatorio | Integr | Derivado |
|---|---|---|---|---|---|---|---|
| material_id | CHAR(5) | FK | No | No | Si | PK,Fk, nulo | No |
| tesis_id | CHAR(10) | No | No | No | Si | No | No |
| carreraTema | VARCHAR2(20) | No | No | No | Si | No | No |
| anoPublicacion, | VARCHAR2(4) | No | No | No | Si | No | No |
| director_id | CHAR(10) | FK | No | No | Si | Fk, nulo | No |

EJEMPLAR

| Nombre | Tipo de Dato | Llave | Rest Dom | Omisión | Obligatorio | Integridad | Derivado |
|---|---|---|---|---|---|---|---|
| noEjemplar | CHAR (10) | Pk | No | No | Sí | Pk, nulo | No |
| material_id | CHAR (5) | Fk | No | No | Sí | Fk, nulo | No |
| estatus_id | VARCHAR2(20) | Fk | No | No | Sí | Fk, nulo | No |

TIPOLECTOR

| Nombre | Tipo de Dato | Llave | Rest Dom | Omisión | Obligatorio | Integridad | Derivado |
|---|---|---|---|---|---|---|---|
| tipolector_id | CHAR (2) | Pk | No | No | Sí | Pk | No |
| descripcionTipoLector | VARCHAR2(20) | No | No | No | Sí | No | No |
| limiteDeMateriales | CHAR (5) | No | No | No | Sí | No | No |
| refrendos | CHAR (5) | No | No | No | Sí | No | No |
| diasPrestamo | CHAR (5) | No | No | No | Sí | No | No |

LECTOR

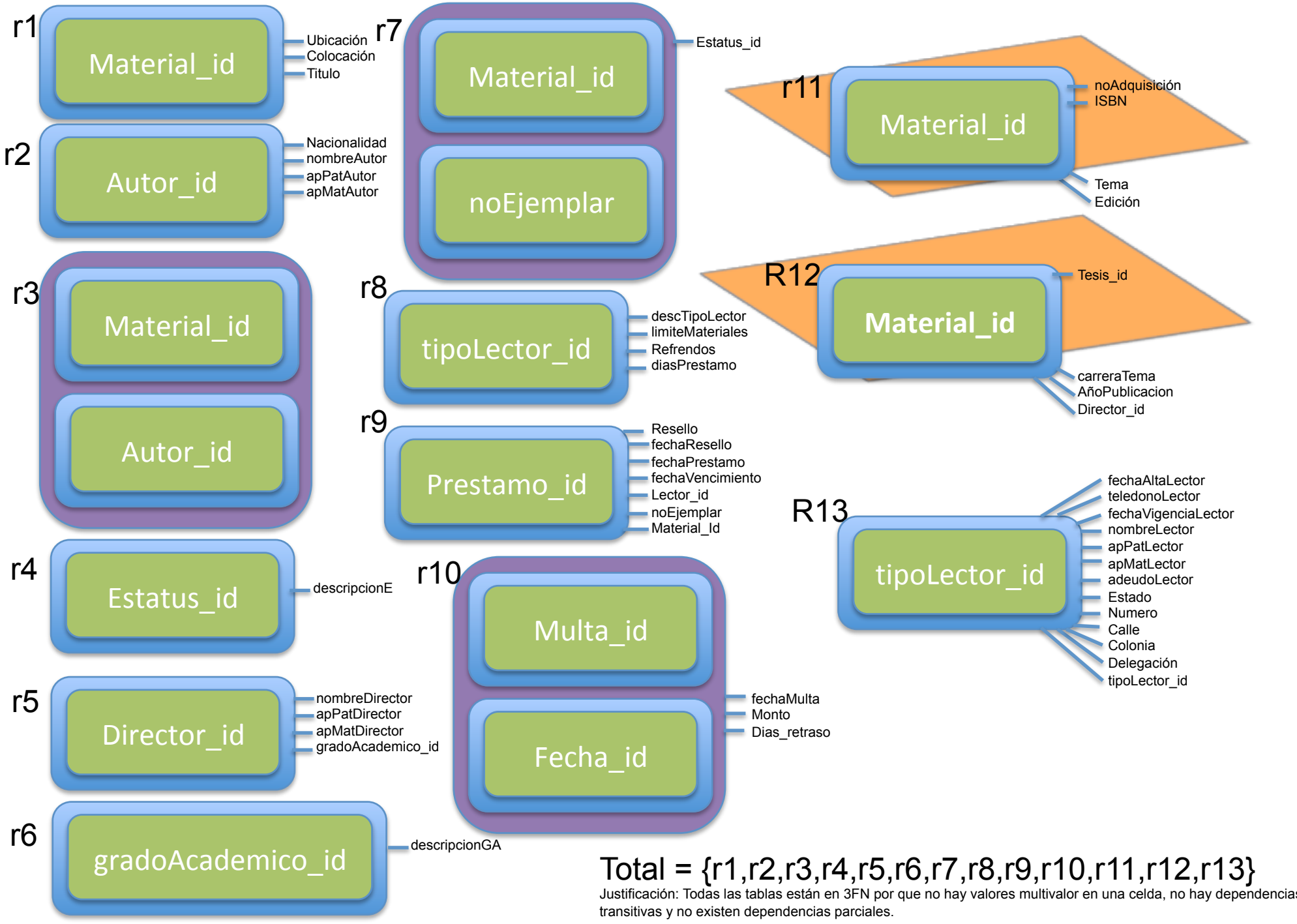
| Nombre | Tipo de Dato | Llave | Rest Dom | Omisión | Obligatorio | Integridad | Derivado |
|---|---|---|---|---|---|---|---|
| lector_id | CHAR (10) | Pk | No | No | Sí | Pk | No |
| fechaAltaLector | DATE | No | No | No | Sí | No | No |
| telefonoLector | VARCHAR2(12) | No | No | No | Sí | No | No |
| fechaVigenciaLector | DATE | No | No | No | Sí | No | No |
| nombreLector | VARCHAR2(20) | No | No | No | Sí | No | No |
| apPaternoLector | VARCHAR2(20) | No | No | No | Sí | No | No |
| apMaternoLector | VARCHAR2(20) | No | No | No | Sí | No | No |
| adeudoLector | NUMBER (8,2) | No | No | No | Sí | No | No |
| estado | VARCHAR2(15) | No | No | No | Sí | No | No |
| numoer | VARCHAR2(8) | No | No | No | Sí | No | No |
| calle | VARCHAR2(15) | No | No | No | Sí | No | No |
| colonia | VARCHAR2(15) | No | No | No | Sí | No | No |
| delegacion | VARCHAR2(20) | No | No | No | Sí | No | No |
| tipolector_id | CHAR(3) | Fk | No | No | Sí | Fk, nulo | No |

PRESTAMO

| Nombre | Tipo de Dato | Llave | Rest Dom | Omisión | Obligatorio | Integridad | Derivado |
|---|---|---|---|---|---|---|---|
| prestamo_id | CHAR(5) | Pk | No | No | Sí | No | No |
| resello | NUMBER | No | No | No | Sí | PK | No |
| fechaResello | DATE | No | No | No | Sí | No | No |
| fechaPrestamo | DATE | No | No | SYSDATE | Sí | No | No |
| fechaVencimiento | DATE | No | No | No | Sí | No | No |
| lector_id | CHAR(10) | Fk | No | No | Sí | Fk, nulo | No |
| noEjemplar | CHAR(10) | Fk | No | No | Sí | Fk, nulo | No |
| material_id | CHAR(5) | FK | No | No | Sí | Fk, nulo | No |

MULTA

| Nombre | Tipo de Dato | Llave | Rest Dom | Omisión | Obligatorio | Integridad | Derivado |
|---|---|---|---|---|---|---|---|
| multa_id | CHAR(10) | Pk | No | No | Sí | Pk | No |
| lector_id | CHAR(10) | FK | No | No | Sí | Fk, nulo | No |
| fechaMulta | DATE | No | No | SYSDATE | Sí | No | No |
| monto | NUMBER(10,2) | No | No | No | No | No | No |
| diasRetraso | CHAR(6) | No | No | No | No | No | No |




Total = {r1,r2,r3,r4,r5,r6,r7,r8,r9,r10,r11,r12,r13}

Justificación: Todas las tablas están en 3FN por que no hay valores multivalor en una celda, no hay dependencias transitivas y no existen dependencias parciales.

# DDL

```
SQL> CONNECT proyecto/proyecto
Connected.
SQL> SET SERVEROUTPUT ON
SQL>
SQL> --Desde usuario "proyecto"
SQL> CREATE TABLE autor(
  2    autor_id CHAR(10) NOT NULL,
  3    nombreAutor VARCHAR2(20) NOT NULL,
  4    apPaternoAutor VARCHAR2(20) NOT NULL,
  5    apMaternoAutor VARCHAR2(20) NULL,
  6    nacionalidad VARCHAR2(15) NOT NULL,
  7    CONSTRAINT Pkautor PRIMARY KEY (autor_id)
  8    );

Table created.

SQL>
SQL> CREATE TABLE material(
  2    material_id CHAR(5) NOT NULL,
  3    ubicacion VARCHAR2(20) NOT NULL,
  4    colocacion VARCHAR2(20) NOT NULL,
  5    titulo VARCHAR2(100) NOT NULL,
  6    tipoMaterial CHAR(1) NOT NULL,
  7    CONSTRAINT PkMaterial PRIMARY KEY (material_id),
  8    CONSTRAINT CktipoMaterial CHECK (tipoMaterial = 'L' OR tipoMaterial = 'T')
  9    );

Table created.

SQL>
SQL> CREATE TABLE escribe(
  2    material_id CHAR(5) NOT NULL,
  3    autor_id CHAR(10) NOT NULL,
  4    CONSTRAINT Pkescribe PRIMARY KEY (material_id, autor_id),
  5    CONSTRAINT Fkmaterial_id FOREIGN KEY (material_id)
  6    REFERENCES material ON DELETE SET NULL,
  7    CONSTRAINT Fkautor_id FOREIGN KEY (autor_id)
  8    REFERENCES autor ON DELETE SET NULL
  9    );

Table created.

SQL>
SQL> CREATE TABLE libro(
  2    material_id CHAR(5) NOT NULL,
  3    noAdquisicion CHAR(10) NOT NULL,
  4    ISBN CHAR(16) NOT NULL,
  5    tema VARCHAR2(50) NOT NULL,
  6    edicion VARCHAR2(20) NOT NULL,
  7    CONSTRAINT PkLibro PRIMARY KEY (material_id),
  8    CONSTRAINT FkLibro FOREIGN KEY (material_id)
  9    REFERENCES material ON DELETE SET NULL,
 10    CONSTRAINT AkLibro1 UNIQUE(noAdquisicion),
 11    CONSTRAINT AkLibro2 UNIQUE (ISBN)
 12    );

Table created.

SQL>
SQL> CREATE TABLE gradoAcademico(
  2    gradoAcademico_id VARCHAR2(15) NOT NULL,
  3    descripcionGA VARCHAR2(20) NOT NULL,
  4    CONSTRAINT PkgradoAcademico PRIMARY KEY (gradoAcademico_id)
```

```
  5    );

Table created.

SQL>
SQL> CREATE TABLE directorTesis(
  2    director_id CHAR(10) NOT NULL,
  3    nombreDirector VARCHAR2(20) NOT NULL,
  4    apPaternoDirector VARCHAR2(20) NOT NULL,
  5    apMaternoDirector VARCHAR2(20) NULL,
  6    gradoAcademico_id VARCHAR2(15) NOT NULL,
  7    CONSTRAINT PkdirectorTesis PRIMARY KEY (director_id),
  8    CONSTRAINT FkgradoAcademico FOREIGN KEY (gradoAcademico_id)
  9    REFERENCES gradoAcademico ON DELETE SET NULL
 10    );

Table created.

SQL>
SQL>
SQL> CREATE TABLE tesis(
  2    material_id CHAR(5) NOT NULL,
  3    tesis_id CHAR(10) NOT NULL,
  4    carreraTema VARCHAR2(20) NOT NULL,
  5    anoPublicacion VARCHAR2(4) NOT NULL,
  6    director_id CHAR(10) NULL,
  7    CONSTRAINT PkTesis PRIMARY KEY (material_id),
  8    CONSTRAINT FkTesis FOREIGN KEY (material_id)
  9    REFERENCES material ON DELETE SET NULL,
 10    CONSTRAINT Fktesis_id FOREIGN KEY (director_id)
 11    REFERENCES directorTesis ON DELETE SET NULL
 12    );

Table created.

SQL>
SQL> CREATE TABLE estatus(
  2    estatus_id VARCHAR2(20) NOT NULL,
  3    descripcionE VARCHAR2(20) NOT NULL,
  4    CONSTRAINT Pkestatus PRIMARY KEY (estatus_id)
  5    );

Table created.

SQL>
SQL> CREATE TABLE ejemplar(
  2    noEjemplar CHAR(10) NOT NULL,
  3    material_id CHAR(5) NOT NULL,
  4    estatus_id VARCHAR2(20) NOT NULL,
  5    CONSTRAINT Pkejemplar PRIMARY KEY (noEjemplar, material_id),
  6    CONSTRAINT Fkmaterial FOREIGN KEY (material_id)
  7    REFERENCES material ON DELETE SET NULL,
  8    CONSTRAINT Fkestatus_id FOREIGN KEY (estatus_id)
  9    REFERENCES estatus ON DELETE SET NULL
 10    );

Table created.

SQL>
SQL> CREATE TABLE tipoLector(
  2    tipolector_id CHAR(3) NOT NULL,
  3    descripcionTipoLector VARCHAR2(20) NOT NULL,
  4    limiteDeMateriales CHAR(5) NOT NULL,
```

```
  5    refrendos CHAR(5) NOT NULL,
  6    diasPrestamo CHAR(5) NOT NULL,
  7    CONSTRAINT PktipoLector PRIMARY KEY (tipolector_id)
  8    );

Table created.

SQL>
SQL> CREATE TABLE lector(
  2    lector_id CHAR(10) NOT NULL,
  3    fechaAltaLector DATE NOT NULL,
  4    telefonoLector VARCHAR2(12) NOT NULL,
  5    fechaVigenciaLector DATE NOT NULL,
  6    nombreLector VARCHAR2(20) NOT NULL,
  7    apPaternoLector VARCHAR2(20) NOT NULL,
  8    apMaternoLector VARCHAR2(20) NULL,
  9    adeudoLector NUMBER(8,2) NOT NULL,
 10    estado VARCHAR2(15) NOT NULL,
 11    numero VARCHAR2(8) NOT NULL,
 12    calle VARCHAR2(15) NOT NULL,
 13    colonia VARCHAR2(15) NOT NULL,
 14    delegacion VARCHAR2(20) NOT NULL,
 15    tipolector_id CHAR(3) NOT NULL,
 16    CONSTRAINT Pklector PRIMARY KEY (lector_id),
 17    CONSTRAINT FktipoLector_id FOREIGN KEY (tipolector_id)
 18    REFERENCES tipoLector ON DELETE SET NULL
 19    );

Table created.

SQL>
SQL> CREATE TABLE prestamo(
  2    prestamo_id CHAR(5) NOT NULL,
  3    resello NUMBER NOT NULL,
  4    fechaResello DATE NULL,
  5    fechaPrestamo DATE DEFAULT SYSDATE NOT NULL,
  6    fechaVencimiento DATE NOT NULL,
  7    lector_id CHAR(10) NOT NULL,
  8    noEjemplar CHAR(10) NOT NULL,
  9    material_id CHAR(5) NOT NULL,
 10    CONSTRAINT Pkprestamo PRIMARY KEY (prestamo_id),
 11    CONSTRAINT Fklector FOREIGN KEY (lector_id)
 12    REFERENCES lector ON DELETE SET NULL,
 13    CONSTRAINT Fkejemplar FOREIGN KEY (noEjemplar, material_id)
 14    REFERENCES ejemplar ON DELETE SET NULL
 15    );

Table created.

SQL>
SQL> CREATE TABLE prestamo_aux(
  2    prestamo_id CHAR(5) NOT NULL,
  3    resello NUMBER NOT NULL,
  4    fechaResello DATE NULL,
  5    fechaPrestamo DATE DEFAULT SYSDATE NOT NULL,
  6    fechaVencimiento DATE NOT NULL,
  7    lector_id CHAR(10) NOT NULL,
  8    noEjemplar CHAR(10) NOT NULL,
  9    material_id CHAR(5) NOT NULL
 10    );

Table created.
```

```
SQL>
SQL> CREATE TABLE multa(
  2    multa_id CHAR(10) NOT NULL,
  3    lector_id CHAR(10) NOT NULL,
  4    fechaMulta DATE DEFAULT SYSDATE NOT NULL,
  5    monto NUMBER(10,2),
  6    diasRetraso CHAR(6),
  7    CONSTRAINT Pkmulta PRIMARY KEY (multa_id),
  8    CONSTRAINT FkLectorId FOREIGN KEY (lector_id)
  9    REFERENCES lector(lector_id) ON DELETE set null
 10    );

Table created.

SQL> spool off;

**********************************************************************************************************
SQL> create sequence SeqAltaMulta
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL>
SQL> --Procedmineto AltaMaterial
SQL> create sequence SeqAltaMaterial
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL>
SQL> --sequencia AltaLibro
SQL> create sequence SeqAltalibro
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL>
SQL>   --Procedmineto AltaTesis, para el tesis_id
SQL> create sequence SeqAltaTesis
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL>
SQL> --Procedmineto AltaGradoAcademico
SQL> create sequence SeqAltaGradoAcademico
  2    start with 0
```

# Secuencias

```
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL>
SQL> create sequence SeqAltaPrestamo
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL> --secuencia para alta de autor
SQL> create sequence SeqAltaAutor
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL> --secuencia director tesis
SQL> create sequence SeqAltaDirectorTesis
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL> --secuencia altaejemplar
SQL> create sequence SeqAltaEjemplar
  2    start with 0
  3    increment by 1
  4    maxvalue 1000
  5    minvalue 0
  6    nocycle;

Sequence created.

SQL> spool off;
```

# Procedimiento

```
**********************************************************************************************************
SQL> create or replace procedure AltaLibro(
  2    --parametros para la tabla material
  3    vUbicacion in varchar2,
  4    vColocacion in varchar2,
  5    vTitulo in varchar2,
  6    vTipo in char,
  7    --parametro para agregar el autor en la tabla escribe
  8    --tomamos en cuenta que el autor existe
  9    vAutor_id in char,
 10    --parametros para agregar a la tabla libro
 11    vISBN in char,
 12    vTema in varchar2,
 13    vEdicion in varchar2
 14  )
```

```
 15  is
 16    vMaterial_id char(5);
 17    vCantidadDeAutores number;
 18    vCantidadDeLibros number;
 19    vCantidadDeAdquisiciones number;
 20  begin
 21    if vTipo != 'L' then
 22      raise_application_error(-20049,'ERROR El tipo de material no es libro, ocupa
 23        el otro procedimiento para agregar tesis');
 24    end if;
 25    --se cuenta cuantos autores con ese Id existen
 26    select count(*)
 27    into vCantidadDeAutores
 28    from autor
 29    where autor_id=vAutor_id;
 30    --se valida que el autor_id exista dentro de la base de datos, en la tabla
 31    --pertinente
 32    if vCantidadDeAutores=1 then
 33      --se encuentra el id del nuevo material
 34      select count(*)
 35      into vCantidadDeLibros
 36      from material;
 37
 38      vCantidadDeAdquisiciones:=SeqAltalibro.nextval;
 39
 40      --se asigna a la variable pertinente
 41      vMaterial_id:= 'M' || SeqAltaMaterial.NEXTVAL;
 42      --se inserta en la tabla material
 43      insert into material (material_id,ubicacion,colocacion,titulo,tipoMaterial)
 44      values(vMaterial_id,vUbicacion,vColocacion,vTitulo,vTipo);
 45      --se inserta en la tabla escribe
 46      insert into escribe (material_id,autor_id)
 47      values(vMaterial_id,vAutor_id);
 48      --se inserta en la tabla libro
 49      insert into libro (material_id,noAdquisicion,ISBN,tema,edicion)
 50      values(vMaterial_id,vCantidadDeAdquisiciones,vISBN,vTema,vEdicion);
 51
 52      dbms_output.put_line('Se dio de alta exitosamente al libro '|| vMaterial_id);
 53
 54    else
 55      raise_application_error(-20050,'ERROR No existe un autor con ese Id,
 56        favor de dar de alta ese autor antes de generar un material a su nombre');
 57    end if;
 58
 59  end;
 60  /

Procedure created.

SQL>
SQL> /*BorraLibro*/

Procedure created.

SQL> create or replace procedure BorraLibro(
  2    vMaterial_id char
  3  )
  4  is
  5    vPrestamos number;
  6    vEjemplares number;
  7    vBandera number;
  8  begin
  9    --se cuenta cuantos prestamos hay de ese libro
```

```
 10    select count(*)
 11    into vPrestamos
 12    from prestamo
 13    where material_id=vMaterial_id;
 14    --se cuenta cuantos ejemplares hay de ese libro
 15    select count(*)
 16    into vEjemplares
 17    from ejemplar
 18    where material_id=vMaterial_id;
 19    --se compara si existen prestamos del libro, si existen prestamos no se puede
 20    --eliminar
 21    if vPrestamos != 0 then
 22      raise_application_error(-20051,'ERROR No puede eliminar ese libro mientras
 23        existan ejemplares del libro en prestamo, se encontraron '
 24        ||vPrestamos
 25        ||' prestamos');
 26    else
 27      --de acuerdo a la cantidad de ejemplares del libro, si es diferente de 0
 28      --se le pedir  al usuario que confirme si quiere eliminar o no
 29      --el libro con todos los ejemplares
 30      if vEjemplares!=0 then
 31        dbms_output.put_line('Existen '
 32          ||vEjemplares
 33          ||' ejemplares de ese libro, se borraran junto con el libro');
 34
 35      end if;
 36      --se borrara el libro junto
 37      --con lo que contemple, borrar los ejemplares, su relacion en escribe,
 38      --su registro en libro y en material
 39
 40      delete from ejemplar where material_id=vMaterial_id;
 41      delete from libro where material_id=vMaterial_id;
 42      delete from escribe where material_id=vMaterial_id;
 43      delete from material where material_id=vMaterial_id;
 44
 45    end if;
 46
 47  end;
 48  /

Procedure created.

SQL> /*ActualizaLibro*/

Procedure created.

SQL> create or replace procedure ActualizaLibro(
  2    --se indican tres parametros, el id del material libro a actualizar
  3    --el campo que se actualiza y el valor del campo
  4    vMaterial_id in char,
  5    vCampo in varchar2,
  6    vValor in varchar2
  7  )
  8  is
  9  begin
 10    --mediante un case checamos los posibles valores que puede tener 'vCampo'
 11    case
 12      when upper(vCampo) = 'UBICACION' then
 13        update material set ubicacion=vValor
 14        where material_id = vMaterial_id;
 15
 16      when upper(vCampo) = 'COLOCACION' then
 17        update material set colocacion=vValor
```

```
 18        where material_id = vMaterial_id;
 19
 20      when upper(vCampo) = 'TITULO' then
 21        update material set titulo=vValor
 22        where material_id = vMaterial_id;
 23
 24      when upper(vCampo) = 'NOADQUISICION' then
 25        update libro set noAdquisicion=vValor
 26        where material_id = vMaterial_id;
 27
 28      when upper(vCampo) = 'ISBN' then
 29        update libro set ISBN=vValor
 30        where material_id = vMaterial_id;
 31
 32      when upper(vCampo) = 'TEMA' then
 33        update libro set tema=vValor
 34        where material_id = vMaterial_id;
 35
 36      when upper(vCampo) = 'EDICION' then
 37        update libro set edicion=vValor
 38        where material_id = vMaterial_id;
 39
 40      else
 41        raise_application_error(-20054,'ERROR No existe ese campo');
 42
 43    end case;
 44  end;
 45  /

Procedure created.

SQL> -------------------------------------------2.-TESIS. (Dar de alta el MATERIAL). - Oscar
SQL> /*AltaTesis*/

Procedure created.

SQL> set serveroutput on
SQL> CREATE OR REPLACE PROCEDURE AltaTesis(
  2    -- DATOS PARA TESIS
  3    v_carreraTema     IN tesis.carreraTema%TYPE,
  4    v_anoPublicacion  IN tesis.anoPublicacion%TYPE,
  5    v_director_id     IN tesis.director_id%TYPE, -- YA DEBE DE EXISTIR EL DIRECTOR
  6    -- DATOS PARA MATERIAL
  7    v_ubicacion       IN material.ubicacion%TYPE,
  8    v_colocacion      IN material.colocacion%TYPE,
  9    v_titulo          IN material.titulo%TYPE,
 10    -- DATOS PARA ESCRIBE .. YA DEBE DE EXISTIR EL AUTOR
 11    v_autor_id        IN autor.autor_id%TYPE
 12  )
 13  AS
 14    v_existeAutor NUMBER;
 15    v_existeDirector NUMBER;
 16    v_material_id material.material_id%TYPE;
 17    v_tesis_id tesis.tesis_id%TYPE;
 18  BEGIN
 19
 20    --VERIFIACR SI EXITE EL MATERIAL_ID
 21    SELECT COUNT(*) INTO v_existeAutor -- si es 0 no existe
 22    FROM autor
 23    WHERE autor_id = v_autor_id;
 24
 25    SELECT COUNT(*) INTO v_existeDirector -- si es 0 no existe
 26    FROM directorTesis
```

```
 27    WHERE director_id = v_director_id;
 28
 29    IF v_existeAutor = 0 OR v_existeDirector = 0 THEN  --SI NO EXISTE AUTOR
 30      raise_application_error(-20100,'ERROR Debe de existir autor y director');
 31    ELSE
 32      v_material_id := 'M' || SeqAltaMaterial.NEXTVAL;
 33      v_tesis_id := 'T' || SeqAltaTesis.NEXTVAL;
 34
 35      INSERT INTO material
 36      VALUES(v_material_id, v_ubicacion, v_colocacion, v_titulo, 'T');
 37
 38      INSERT INTO escribe
 39      VALUES(v_material_id, v_autor_id);
 40
 41      INSERT INTO tesis
 42      VALUES (v_material_id, v_tesis_id, v_carreraTema, v_anoPublicacion, v_director_id);
 43      DBMS_OUTPUT.PUT_LINE('Se inserto un nuevo material tipo tesis:  ' || v_material_id);
 44      COMMIT;
 45    END IF;
 46  END AltaTesis;
 47  /

Procedure created.

SQL> /*BajaTesis*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE BajaTesis(
  2    v_material_id     IN tesis.material_id%TYPE
  3  )
  4  AS
  5    v_existePrestamo NUMBER;
  6  BEGIN
  7
  8    SELECT COUNT(*) INTO v_existePrestamo
  9    FROM prestamo
 10    WHERE material_id = v_material_id;
 11
 12    IF v_existePrestamo = 0 THEN
 13    DELETE FROM ejemplar  WHERE material_id = v_material_id;
 14    DELETE FROM tesis     WHERE material_id = v_material_id;
 15    DELETE FROM escribe   WHERE material_id = v_material_id;
 16    DELETE FROM material  WHERE material_id = v_material_id;
 17    COMMIT;
 18    DBMS_OUTPUT.PUT_LINE('Se elimino el material tipo tesis con id:  ' || v_material_id);
 19    ELSE
 20      raise_application_error(-20101,'ERROR La tesis no debe estar en prestamo');
 21    END IF;
 22
 23  END BajaTesis;
 24  /

Procedure created.

SQL>
SQL> /*ActualizaTesis*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE ActualizaTesis(
  2    vMaterial_id IN material.material_id%TYPE,
  3    vCampo IN VARCHAR2,
```

```
  4    vValor IN VARCHAR2
  5  )
  6  is
  7  BEGIN
  8    CASE
  9      WHEN UPPER(vCampo) = 'UBICACION' THEN
 10        UPDATE material SET ubicacion=vValor
 11        WHERE material_id = vMaterial_id;
 12
 13      WHEN UPPER(vCampo) = 'COLOCACION' THEN
 14        UPDATE material SET colocacion=vValor
 15        WHERE material_id = vMaterial_id;
 16
 17      WHEN UPPER(vCampo) = 'TITULO' THEN
 18        UPDATE material SET titulo=vValor
 19        WHERE material_id = vMaterial_id;
 20
 21      WHEN UPPER(vCampo) = 'TESIS_ID' THEN
 22        UPDATE tesis SET tesis_id=vValor
 23        WHERE material_id = vMaterial_id;
 24
 25      WHEN UPPER(vCampo) = 'CARRERATEMA' THEN
 26        UPDATE tesis SET carreraTema=vValor
 27        WHERE material_id = vMaterial_id;
 28
 29      WHEN UPPER(vCampo) = 'ANOPUBLICACION' THEN
 30        UPDATE tesis SET anoPublicacion=vValor
 31        WHERE material_id = vMaterial_id;
 32
 33      WHEN UPPER(vCampo) = 'DIRECTOR_ID' THEN
 34        UPDATE tesis SET director_id=vValor
 35        WHERE material_id = vMaterial_id;
 36
 37      else
 38        raise_application_error(-20103,'ERROR No existe ese campo');
 39
 40    end case;
 41  end;
 42  /

Procedure created.

SQL>
SQL> ------------------------------------------3.-DIRECTOR DE TESIS.--Chavira
SQL> /*AltaDirTesis*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE AltaDirTesis(
  2    vDirector_id IN directorTesis.director_id%TYPE,
  3    -- Se ingresar  la descripci¢n del Grado Acad,mico. Se buscar  el id.
  4    vGradoAcademicoDescrip IN gradoAcademico.descripcionGA%TYPE,
  5    -- Se mandan al final los par metros para poder admitir un Apellido
  6    -- Materno nulo al final (no se manda valor en la ejecuci¢n).
  7    vNombreDirector IN directorTesis.nombreDirector%TYPE,
  8    vApPaternoDirector IN directorTesis.apPaternoDirector%TYPE,
  9    vApMaternoDirector IN directorTesis.apMaternoDirector%TYPE DEFAULT NULL
 10  )
 11  AS
 12  vBuscaGradoAcademico directorTesis.gradoAcademico_id%TYPE;
 13  vDT_id CHAR(5);
 14  BEGIN
 15  vDT_id := 'D' || vDirector_id;
```

```
 16  SELECT gradoAcademico_id INTO vBuscaGradoAcademico
 17  FROM gradoAcademico WHERE descripcionGA=vGradoAcademicoDescrip;
 18  IF (vBuscaGradoAcademico IS NOT NULL) THEN
 19    INSERT INTO directorTesis
 20    VALUES(vDT_id, vNombreDirector, vApPaternoDirector, vApMaternoDirector, vBuscaGradoAcademico);
 21    DBMS_OUTPUT.PUT_LINE('Se insert¢ al director de tesis con id: '|| vDT_id);
 22  ELSE
 23    DBMS_OUTPUT.PUT_LINE('No se encontr¢ registrado el grado de ' ||vGradoAcademicoDescrip);
 24  END IF;
 25  COMMIT;
 26  END AltaDirTesis;
 27  /

Procedure created.

SQL> /* EJECUCION:
SQL> EXEC AltaDirTesis(director_id, gradoEnTexto, Nombre, ApPat, ApMat);
SQL> - gradoEnTexto, por ejemplo: 'Kinder', 'Secu', de la tabla gradoAcademico
SQL> - apMat se puede ignorar:
SQL> EXEC AltaDirTesis(director_id, gradoEnTexto, Nombre, ApPat); */
SQL>
SQL> /*BajaDirTesis*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE BajaDirTesis(
  2    vDirector_id directorTesis.director_id%TYPE
  3  )
  4  AS
  5  vBuscaGradoAcademico directorTesis.gradoAcademico_id%TYPE;
  6  BEGIN
  7  DELETE FROM directorTesis WHERE director_id=vDirector_id;
  8  DBMS_OUTPUT.PUT_LINE('Se elimin¢ al director de tesis con id: ' ||vDirector_id);
  9  COMMIT;
 10  END BajaDirTesis;
 11  /

Procedure created.

SQL>
SQL> /*ActualizaDirTesis*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE ActualizaDirTesis(
  2    vDirector_id IN directorTesis.director_id%TYPE,
  3    -- Se ingresar  la descripci¢n del Grado Acad,mico. Se buscar  el id.
  4    vGradoAcademicoDescrip IN gradoAcademico.descripcionGA%TYPE,
  5    -- Se mandan al final los par metros para poder admitir un Apellido
  6    -- Materno nulo al final (no se manda valor en la ejecuci¢n).
  7    vNombreDirector IN directorTesis.nombreDirector%TYPE,
  8    vApPaternoDirector IN directorTesis.apPaternoDirector%TYPE,
  9    vApMaternoDirector IN directorTesis.apMaternoDirector%TYPE DEFAULT NULL
 10  )
 11  AS
 12  vBuscaGradoAcademico directorTesis.gradoAcademico_id%TYPE;
 13  BEGIN
 14  SELECT gradoAcademico_id INTO vBuscaGradoAcademico
 15  FROM gradoAcademico WHERE descripcionGA=vGradoAcademicoDescrip;
 16  IF (vBuscaGradoAcademico IS NOT NULL) THEN
 17    UPDATE directorTesis SET
 18      nombreDirector=vNombreDirector,
 19      apPaternoDirector=vApPaternoDirector,
```

```
 20      apMaternoDirector=vApMaternoDirector,
 21      gradoAcademico_id=vBuscaGradoAcademico
 22    WHERE director_id=vDirector_id;
 23    DBMS_OUTPUT.PUT_LINE('Se insert¢ al director de tesis con id: '|| vDirector_id);
 24  ELSE
 25    DBMS_OUTPUT.PUT_LINE('No se encontr¢ registrado el grado de ' ||vGradoAcademicoDescrip);
 26  END IF;
 27  --COMMIT;
 28  END ActualizaDirTesis;
 29  /

Procedure created.

SQL>
SQL> ----------------------------------------4.-EJEMPLAR.--Joya
SQL> /*AltaEjemplar*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE AltaEjemplar(
  2    vmaterial_id IN CHAR
  3  )
  4  AS
  5    vnoEjemplar CHAR(8);
  6    vCharejemplar CHAR(5);
  7    vmaterial NUMBER;
  8  BEGIN
  9  SELECT count(*) INTO vmaterial
 10  FROM material
 11  WHERE material_id = vmaterial_id;
 12  IF (vmaterial > 0) THEN
 13    SELECT TO_CHAR(TO_NUMBER(SUBSTR(MAX(noEjemplar), 3, 10))+1)
 14    INTO vnoEjemplar
 15    FROM ejemplar
 16    WHERE MATERIAL_ID = vmaterial_id;
 17    IF vnoEjemplar IS NOT NULL THEN
 18      INSERT INTO ejemplar VALUES ('EJ'|| vnoEjemplar, vmaterial_id, 'ES1');
 19    ELSE
 20      INSERT INTO ejemplar VALUES ('EJ0'|| vnoEjemplar, vmaterial_id, 'ES1');
 21    END IF;
 22    COMMIT;
 23  ELSE
 24    DBMS_OUTPUT.PUT_LINE('No existe ningun material registrado aun');
 25  END IF;
 26  END AltaEjemplar;
 27  /

Procedure created.

SQL>
SQL> /*BajaEjemplar*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE BajaEjemplar(
  2    vnoEjemplar IN ejemplar.noEjemplar%TYPE,
  3    vmaterial_id IN ejemplar.material_id%TYPE
  4  )
  5  IS
  6    vestatus_id ejemplar.estatus_id%TYPE;
  7  BEGIN
  8  SELECT estatus_id INTO vestatus_id
  9  FROM ejemplar WHERE (vnoEjemplar = noEjemplar AND vmaterial_id = material_id);
```

```
 10  IF (vestatus_id <> 'ES2') THEN
 11    DELETE FROM ejemplar WHERE (vnoEjemplar = noEjemplar);
 12    DBMS_OUTPUT.PUT_LINE('Se elimin¢ el ejemplar con id: ' || vnoEjemplar);
 13    COMMIT;
 14  ELSE
 15    DBMS_OUTPUT.PUT_LINE('El ejemplar con id ' || vnoEjemplar||' est  en prestamo (No se puede
eliminar)');
 16  END IF;
 17  END BajaEjemplar;
 18  /

Procedure created.

SQL>
SQL> /*ActualizaEjemplar*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE ActualizaEjemplar(
  2    vnoEjemplar IN ejemplar.noEjemplar%TYPE,
  3    vMaterial_id IN ejemplar.material_id%TYPE,
  4    vCampo in varchar2,
  5    vValor in varchar2
  6  )
  7  AS
  8    vEstatusActual CHAR(10);
  9  BEGIN
 10    CASE
 11      when upper(vCampo) = 'NOEJEMPLAR' then
 12        UPDATE ejemplar SET  noEjemplar = vValor
 13        WHERE vnoEjemplar = noEjemplar AND vMaterial_id = material_id;
 14      when upper(vCampo) = 'MATERIAL_ID' then
 15        UPDATE ejemplar SET  material_id = vValor
 16        WHERE vnoEjemplar = noEjemplar AND vMaterial_id = material_id;
 17      when upper(vCampo) = 'ESTATUS_ID' then
 18        UPDATE ejemplar SET  estatus_id = vValor
 19        WHERE vnoEjemplar = noEjemplar AND vMaterial_id = material_id;
 20    ELSE
 21      raise_application_error(-20051,'ERROR No existe ese campo, o usted no lo puede modificar');
 22    END CASE;
 23    COMMIT;
 24  END ActualizaEjemplar;
 25  /

Procedure created.

SQL> ----------------------------------------------5.-LECTOR. --L zaro
SQL> /*AltaLector*/

Procedure created.

SQL> create or replace procedure AltaLector(
  2    vTelefono in varchar2,
  3    vNombre in varchar2,
  4    vApPaternoLector in varchar2,
  5    vApMaternoLector in varchar2,
  6    vEstado in varchar2,
  7    vNumero in varchar2,
  8    vCalle in varchar2,
  9    vColonia in varchar2,
 10    vDelegacion in varchar2,
 11    vTipoLector_id in char
 12  )
```

```
 13  is
 14    vLector_id char(10);
 15    vCantidadDeTipos number(4,0);
 16    vCantidadDeLectores number(4,0);
 17  begin
 18    select count(*)
 19    into vCantidadDeTipos
 20    from tipolector
 21    where tipolector_id = vTipoLector_id;
 22
 23    select count(*)+1
 24    into vCantidadDeLectores
 25    from lector;
 26
 27    vLector_id:='L'||vCantidadDeLectores;
 28
 29    if vCantidadDeTipos != 0 then
 30      insert into lector (lector_id,fechaAltaLector,telefonoLector,fechaVigenciaLector,
 31        nombreLector,apPaternoLector,apMaternoLector,AdeudoLector,estado,numero,calle,
 32        colonia,delegacion,tipolector_id)
 33      values(vLector_id,sysdate,vTelefono,sysdate+365,
 34        vNombre,vApPaternoLector,vApMaternoLector,0,vEstado,vNumero,vCalle,
 35        vColonia,vDelegacion,vTipoLector_id);
 36      dbms_output.put_line('Alta de Usuario '
 37        ||vLector_id
 38        ||' exitosa');
 39    else
 40      raise_application_error(-20052,'ERROR Ese tipo de lector no existe');
 41    end if;
 42  end;
 43  /

Procedure created.

SQL> show errors
No errors.
SQL>
SQL>
SQL> /*BorraLector*/

Procedure created.

SQL> create or replace procedure BorraLector(
  2    vLector_id in char
  3  )
  4  is
  5    vCantidadDePrestamos number(4,0);
  6  begin
  7    --se cuenta cuantos prestamos tiene el lector
  8    select count(*)
  9    into vCantidadDePrestamos
 10    from prestamo
 11    where lector_id=vLector_id;
 12    --si no tiene prestamos, se puede eliminar, en otro caso no se puede y se
 13    --lanza la excepcion
 14    if vCantidadDePrestamos = 0 then
 15      delete from lector where lector_id=vLector_id;
 16    else
 17      raise_application_error(-20054,'ERROR No se puede eliminar al lector
 18        mientras tenga prestamos en curso, se encontraron '
 19        ||vCantidadDePrestamos
 20        ||' prestamos registrados');
 21    end if;
```

```
 22  end;
 23  /

Procedure created.

SQL> show errors;
No errors.
SQL>
SQL> /*ActualizaLector*/

Procedure created.

SQL> create or replace procedure ActualizaLector(
  2    vLector_id in char,
  3    vCampo in varchar2,
  4    vValor in varchar2
  5  )
  6  is
  7    vFecha date;
  8  begin
  9    case
 10      when upper(vCampo) = 'FECHAALTALECTOR' then
 11
 12        if to_date(vValor,'dd/mm/yy') = SYSDATE then
 13          vFecha := sysdate;
 14        else
 15          vFecha := to_date(vValor,'dd/mm/yy');
 16        end if;
 17
 18        update lector set fechaAltaLector = vFecha,
 19          fechaVigenciaLector = vFecha+365
 20        where lector_id=vLector_id;
 21
 22      when upper(vCampo) = 'TELEFONOLECTOR' then
 23
 24        update lector set telefonoLector=vValor
 25        where lector_id=vLector_id;
 26
 27      when upper(vCampo) = 'NOMBRELECTOR' then
 28
 29        update lector set nombreLector=vValor
 30        where lector_id=vLector_id;
 31
 32      when upper(vCampo) = 'APPATERNOLECTOR' then
 33
 34        update lector set apPaternoLector=vValor
 35        where lector_id=vLector_id;
 36
 37      when upper(vCampo) = 'APMATERNOLECTOR' then
 38
 39        update lector set apMaternoLector=vValor
 40        where lector_id=vLector_id;
 41
 42      when upper(vCampo) = 'ADEUDOLECTOR' then
 43
 44        update lector set adeudoLector=vValor
 45        where lector_id=vLector_id;
 46
 47      when upper(vCampo) = 'ESTADO' then
 48
 49        update lector set estado=vValor
 50        where lector_id=vLector_id;
 51
```

```
 52      when upper(vCampo) = 'NUMERO' then
 53
 54        update lector set numero=vValor
 55        where lector_id=vLector_id;
 56
 57      when upper(vCampo) = 'CALLE' then
 58
 59        update lector set calle=vValor
 60        where lector_id=vLector_id;
 61
 62      when upper(vCampo) = 'COLONIA' then
 63
 64        update lector set colonia=vValor
 65        where lector_id=vLector_id;
 66
 67      when upper(vCampo) = 'DELEGACION' then
 68
 69        update lector set delegacion=vValor
 70        where lector_id=vLector_id;
 71
 72      when upper(vCampo) = 'TIPOLECTOR_ID' then
 73
 74        update lector set tipoLector_id=vValor
 75        where lector_id=vLector_id;
 76
 77      else
 78        raise_application_error(-20054,'ERROR No existe ese campo');
 79
 80    end case;
 81  end;
 82  /

Procedure created.

SQL> show errors
No errors.
SQL> -------------------------------------------6.-PRESTAMO. -- Oscar
SQL> /*AltaPrestamo*/

Procedure created.

SQL> set serveroutput on
SQL> CREATE OR REPLACE PROCEDURE AltaPrestamo(
  2    v_lector_id   IN prestamo.lector_id%TYPE, --YA DEBE DE EXISTIR EL LECTOR
  3    v_noEjemplar  IN prestamo.noEjemplar%TYPE,
  4    v_material_id IN prestamo.material_id%TYPE
  5  )
  6  AS
  7  v_estatus_id ejemplar.estatus_id%TYPE;
  8  v_fechaVigenciaLector lector.fechaVigenciaLector%TYPE;
  9  v_prestamo_id  prestamo.prestamo_id%TYPE;
 10  v_diasPrestamo tipoLector.diasPrestamo%TYPE;
 11  v_limiteDeMateriales NUMBER;
 12  v_numeroDeMateriales NUMBER;
 13  v_fechaPrestamo DATE := SYSDATE;
 14  v_fechaVencimiento DATE;
 15  BEGIN
 16
 17    SELECT estatus_id INTO v_estatus_id
 18    FROM ejemplar
 19    WHERE noEjemplar = v_noEjemplar AND v_material_id = material_id;
 20
 21    SELECT fechaVigenciaLector INTO v_fechaVigenciaLector
```

```
 22    FROM lector
 23    WHERE lector_id = v_lector_id;
 24
 25    IF v_estatus_id = 'ES1' THEN
 26      IF v_fechaVigenciaLector > SYSDATE THEN
 27          SELECT tl.limiteDeMateriales INTO v_limiteDeMateriales
 28          FROM tipoLector tl
 29          JOIN lector l
 30          ON l.tipoLector_id = tl.tipoLector_id
 31          WHERE l.lector_id = v_lector_id;
 32
 33          SELECT COUNT(*) INTO v_numeroDeMateriales
 34          FROM prestamo
 35          WHERE lector_id = v_lector_id;
 36
 37          IF v_limiteDeMateriales > v_numeroDeMateriales THEN
 38
 39            SELECT tl.diasPrestamo INTO v_diasPrestamo
 40            FROM tipoLector tl
 41            JOIN lector l
 42            ON l.tipoLector_id = tl.tipoLector_id
 43            WHERE l.lector_id = v_lector_id;
 44
 45            v_fechaVencimiento := SYSDATE + TO_NUMBER(v_diasPrestamo);
 46            v_prestamo_id := 'P' || SeqAltaPrestamo.NEXTVAL;
 47            INSERT INTO prestamo ( PRESTAMO_ID,RESELLO , FECHAPRESTAMO ,FECHAVENCIMIENTO, LECTOR_ID
,NOEJEMPLAR ,MATERIAL_ID)
 48            VALUES (v_prestamo_id, 0, v_fechaPrestamo, v_fechaVencimiento, v_lector_id, v_noEjemplar,
v_material_id);
 49            DBMS_OUTPUT.PUT_LINE('Se realiz¢ el prestamo del material:  ' || v_material_id|| ' Al lector:
'|| v_lector_id);
 50            COMMIT;
 51          ELSE
 52            raise_application_error(-20106,'ERROR Numero de prestamos excedido');
 53          END IF;
 54      ELSE
 55        raise_application_error(-20105,'ERROR La fecha del lector a vencido');
 56      END IF;
 57    ELSE
 58      raise_application_error(-20104,'ERROR El estatus del ejemplar debe ser disponible');
 59    END IF;
 60  END AltaPrestamo;
 61  /

Procedure created.

SQL>
SQL> /*BajaPrestamo*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE BajaPrestamo(
  2    v_prestamo_id  IN prestamo.prestamo_id%TYPE
  3  )
  4  AS
  5  BEGIN
  6    DELETE FROM prestamo WHERE prestamo_id = v_prestamo_id;
  7    COMMIT;
  8    DBMS_OUTPUT.PUT_LINE('Se ha devuelto el material con prestamo_id:  ' || v_prestamo_id);
  9  END BajaPrestamo;
 10  /

Procedure created.
```

```
SQL>
SQL> /*ActualizaPrestamo*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE ActualizaPrestamo(
  2    vprestamo_id IN CHAR,
  3    vCampo IN varchar2,
  4    vValor IN VARCHAR2
  5  )
  6  is
  7  vFecha DATE;
  8  vRefrendo NUMBER;
  9  v_diasPrestamo NUMBER;
 10  BEGIN
 11
 12    SELECT tl.refrendos, tl.diasPrestamo INTO vRefrendo, v_diasPrestamo
 13    FROM tipoLector tl
 14    JOIN lector l
 15    ON l.tipoLector_id = tl.tipoLector_id
 16    JOIN prestamo p
 17    ON p.lector_id = l.lector_id
 18    WHERE p.prestamo_id = vprestamo_id;
 19
 20    CASE
 21      WHEN UPPER(vCampo) = 'RESELLO' THEN
 22        IF vValor < vRefrendo THEN
 23          UPDATE prestamo SET resello = vValor
 24          WHERE prestamo_id = vprestamo_id; --SE ACTUALIZA EL NUMERO DE RESELLOS
 25          UPDATE prestamo SET fechaResello = SYSDATE
 26          WHERE prestamo_id = vprestamo_id; -- SE ACTUALIZA LA FECHA EN LA QUE SE RESELLO
 27          UPDATE prestamo SET fechaVencimiento = SYSDATE + v_diasPrestamo
 28          WHERE prestamo_id = vprestamo_id; -- SE CALCULA OTRAVEZ LA FECHA DE VENCIMIENTO
 29        ELSE
 30          raise_application_error(-20107,'ERROR Numero de resellos excedido');
 31        END IF;
 32
 33      WHEN UPPER(vCampo) = 'FECHARESELLO' THEN
 34        IF TO_DATE(vValor,'dd/mm/yy') = SYSDATE THEN
 35          vFecha := SYSDATE;
 36        ELSE
 37          vFecha := TO_DATE(vValor,'dd/mm/yy');
 38        END if;
 39
 40        UPDATE prestamo
 41        SET fecharesello = vFecha,
 42            fechaVencimiento = vFecha+5
 43        WHERE prestamo_id = vprestamo_id;
 44
 45      WHEN UPPER(vCampo) = 'FECHAPRESTAMO' THEN
 46
 47        IF TO_DATE(vValor,'dd/mm/yy') = SYSDATE THEN
 48          vFecha := SYSDATE;
 49        ELSE
 50          vFecha := TO_DATE(vValor,'dd/mm/yy');
 51        END if;
 52
 53        UPDATE prestamo
 54        SET fechaPrestamo = vFecha,
 55            fechaVencimiento = vFecha+5
 56            WHERE prestamo_id = vprestamo_id;
 57
```

```
 58      WHEN UPPER(vCampo) = 'LECTOR_ID' THEN
 59        UPDATE prestamo SET lector_id = vValor
 60        WHERE prestamo_id = vprestamo_id;
 61
 62      WHEN UPPER(vCampo) = 'NOEJEMPLAR' THEN
 63        UPDATE prestamo SET noEjemplar = vValor
 64        WHERE prestamo_id = vprestamo_id;
 65
 66      WHEN UPPER(vCampo) = 'MATERIAL_ID' THEN
 67        UPDATE prestamo SET material_id = vValor
 68        WHERE prestamo_id = vprestamo_id;
 69
 70      else
 71        raise_application_error(-20054,'ERROR No existe ese campo');
 72    end case;
 73  end;
 74  /

Procedure created.

SQL>
SQL> CREATE OR REPLACE PROCEDURE resello(
  2    vprestamo_id in prestamo.prestamo_id%TYPE
  3  )
  4  AS
  5    vresello prestamo.resello%TYPE;
  6    vLector_id lector.lector_id%TYPE;
  7    vtipolector CHAR(4);
  8  BEGIN
  9  SELECT resello, lector_id INTO vresello, vLector_id
 10  FROM prestamo
 11  WHERE prestamo_id = vprestamo_id;
 12  SELECT tipolector_id INTO vtipolector
 13  FROM lector
 14  WHERE lector_id = vLector_id;
 15  CASE
 16    WHEN UPPER(vtipolector) = 'TL1' THEN
 17      IF vresello < 1 THEN
 18        UPDATE prestamo
 19        SET fecharesello = SYSDATE, fechaPrestamo = SYSDATE, FECHAVENCIMIENTO = SYSDATE + 8, resello =
resello + 1
 20        WHERE prestamo_id = vprestamo_id;
 21      ELSE
 22        DBMS_OUTPUT.PUT_LINE('Favor de devolver el libro');
 23      END IF;
 24    WHEN UPPER(vtipolector) = 'TL2' THEN
 25      IF vresello < 2 THEN
 26        UPDATE prestamo
 27        SET fecharesello = SYSDATE, fechaPrestamo = SYSDATE, FECHAVENCIMIENTO = SYSDATE + 15, resello =
resello + 1
 28        WHERE prestamo_id = vprestamo_id;
 29      ELSE
 30        DBMS_OUTPUT.PUT_LINE('Favor de devolver el libro');
 31      END IF;
 32  ELSE
 33      IF vresello < 3 THEN
 34        UPDATE prestamo
 35        SET fecharesello = SYSDATE, fechaPrestamo = SYSDATE, FECHAVENCIMIENTO = SYSDATE + 30, resello =
resello + 1
 36        WHERE prestamo_id = vprestamo_id;
 37      ELSE
 38        DBMS_OUTPUT.PUT_LINE('Favor de devolver el libro');
 39      END IF;
```

```
 40  END CASE;
 41  END resello;
 42  /

Procedure created.

SQL>
SQL> -----------------------------------------7.-MULTA. - Chavira
SQL> /*AltaMulta*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE AltaMulta(
  2    vPrestamo prestamo.prestamo_id%TYPE,
  3    vLector_id prestamo.lector_id%TYPE,
  4    vfechaVencimiento prestamo.fechaVencimiento%TYPE
  5  )
  6  AS
  7    vmulta_id CHAR(10);
  8    vdias_retraso NUMBER(15);
  9  BEGIN
 10    IF (vfechaVencimiento < SYSDATE) THEN
 11      vmulta_id := 'MU' || SeqAltamulta.NEXTVAL;
 12      vdias_retraso := SYSDATE-vfechaVencimiento;
 13      INSERT INTO multa
 14      VALUES(vMulta_id, vLector_id, SYSDATE, vdias_retraso * 10, vdias_retraso);
 15      DBMS_OUTPUT.PUT_LINE('Se insert¢ una nueva multa con id: ' || vMulta_id);
 16    ELSE
 17      DBMS_OUTPUT.PUT_LINE('Entregado en tiempo, no se genera multa');
 18    END IF;
 19    COMMIT;
 20  END AltaMulta;
 21  /

Procedure created.

SQL>
SQL> /*BajaMulta*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE BajaMulta(
  2    vMulta_id multa.multa_id%TYPE
  3  )
  4  AS
  5  BEGIN
  6    DELETE FROM multa
  7    WHERE multa_id=vMulta_id;
  8    DBMS_OUTPUT.PUT_LINE('Se elimin¢ la multa con id: ' || vMulta_id);
  9  END BajaMulta;
 10  /

Procedure created.

SQL>
SQL> /*ActualizaMulta*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE ActualizaMulta(
  2      vMulta_id multa.multa_id%TYPE,
  3      vFechaMulta multa.fechaMulta%TYPE,
  4      vdiasRetraso multa.diasretraso%TYPE
```

```
  5  )
  6  AS
  7  BEGIN
  8    UPDATE multa
  9    SET fechaMulta = vFechaMulta, monto = vdiasRetraso*10, diasretraso = vdiasRetraso
 10    WHERE vMulta_id = multa_id;
 11  END ActualizaMulta;
 12  /

Procedure created.

SQL>
SQL>
SQL> -------------------------------------------8.-GRADO ACADEMICO. -- Joya
SQL> /*AltaGradoAcademico*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE AltaGradoAcademico(
  2    vdescripcionGA IN gradoAcademico.descripcionGA%TYPE
  3  )
  4  AS
  5    vgrado_id CHAR(10);
  6  BEGIN
  7    vgrado_id := 'GA' || SeqAltaGradoAcademico.NEXTVAL;
  8    INSERT INTO gradoAcademico
  9    VALUES (vgrado_id, vdescripcionGA);
 10    DBMS_OUTPUT.PUT_LINE('Se inserto un nuevo grado academico con id:  ' || vgrado_id);
 11    COMMIT;
 12  END AltaGradoAcademico;
 13  /

Procedure created.

SQL>
SQL> /*BajaGradoAcademico*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE BajaGradoAcademico(
  2      vgradoAcademico_id IN gradoAcademico.gradoAcademico_id%TYPE
  3  )
  4  AS
  5  BEGIN
  6    DELETE gradoAcademico
  7    WHERE vgradoAcademico_id = gradoAcademico_id;
  8    COMMIT;
  9    DBMS_OUTPUT.PUT_LINE('Se elimino el grado academico con id:  ' || vgradoAcademico_id);
 10  END BajaGradoAcademico;
 11  /

Procedure created.

SQL>
SQL> /*ActualizaGradoAcademico*/

Procedure created.

SQL> CREATE OR REPLACE PROCEDURE ActualizaGradoAcademico(
  2    vgradoAcademico_id IN gradoAcademico.gradoAcademico_id%TYPE,
  3    vdescripcionGA IN gradoAcademico.descripcionGA%TYPE
  4  )
  5  AS
```

```
  6  BEGIN
  7    UPDATE gradoAcademico
  8    SET gradoAcademico_id = vgradoAcademico_id,
  9          descripcionGA = vdescripcionGA
 10    WHERE vgradoAcademico_id = gradoAcademico_id;
 11    COMMIT;
 12    DBMS_OUTPUT.PUT_LINE('Se actualiz¢ el grado academico con id:  ' || vgradoAcademico_id);
 13  END ActualizaGradoAcademico;
 14  /

Procedure created.

SQL> ----------------------------------------EXTRA Procedimiento para dar de alta a un autor
SQL> prompt altaautor
altaautor
SQL> create or replace procedure AltaAutor(
  2    pNombreAutor in VARCHAR2,
  3    pApPaternoAutor in VARCHAR2,
  4    pApMaternoAutor in VARCHAR2,
  5    pNacionalidad in VARCHAR2
  6  )
  7  is
  8    vId char(10);
  9  begin
 10    vId:='A'||to_char(SeqAltaAutor.nextval);
 11    insert into autor(autor_id,nombreAutor,apPaternoAutor,apMaternoAutor,nacionalidad)
 12    values(vId,pNombreAutor,pApPaternoAutor,pApMaternoAutor,pNacionalidad);
 13    dbms_output.put_line('Se di¢ de alta al autor '||vId);
 14  end;
 15  /

Procedure created.

SQL> spool off
**********************************************************************************************************
----------1.- LAZARO         Si un lector tiene una multa, no se le podrá n prestar materiales ni eliminar
lector hasta que la haya liquidado la multa.
SQL> create or replace trigger tgMultaLectorEnLector
  2    before delete
  3    on lector
  4    for each row
  5  declare
  6    PRAGMA AUTONOMOUS_TRANSACTION;
  7    vCantidadDeMultas number(4,0);
  8    cursor cur_multa is
  9      select count(*) as multas
 10      from multa
 11      where lector_id=:old.lector_id;
 12
 13  begin
 14    for r in cur_multa loop
 15      vCantidadDeMultas:=r.multas;
 16    end loop;
 17
 18    if vCantidadDeMultas != 0 then
 19      raise_application_error(-20056,'ERROR No se puede dar de baja al
 20        usuario indicado dado que se encontraron '
 21        ||vCantidadDeMultas
 22        ||' multas registradas a su perfil');
 23    end if;
 24  end;
 25  /
```

# Triggers

```
Trigger created.

SQL> show errors;
No errors.
SQL>
SQL> create or replace trigger tgMultaLectorEnPrestamo
  2    before insert
  3    on prestamo
  4    for each row
  5  declare
  6    vCantidadDeMultas number(4,0);
  7  begin
  8    select count(*)
  9    into vCantidadDeMultas
 10    from multa
 11    where lector_id=:new.lector_id;
 12
 13    if vCantidadDeMultas != 0 then
 14      raise_application_error(-20055,'ERROR No se pueden realizar prestamos al
 15        usuario indicado dado que se encontraron '
 16        ||vCantidadDeMultas
 17        ||' multas registradas a su perfil');
 18    end if;
 19
 20  end;
 21  /

Trigger created.

SQL> show errors;
No errors.
SQL>
SQL>
SQL> -----------2.- OSCAR          Al realizarse el pr,stamo de un ejemplar, se deber  de modificar su
estatus autom ticamente. -- Oscar
SQL>
SQL> CREATE OR REPLACE TRIGGER tgPrestamoEjemplar
  2  BEFORE INSERT
  3  ON prestamo
  4  FOR EACH ROW
  5  DECLARE
  6    vEstatus_id ejemplar.estatus_id%TYPE;
  7  BEGIN
  8    SELECT estatus_id INTO vEstatus_id
  9    FROM ejemplar
 10    WHERE noEjemplar = :NEW.noEjemplar AND material_id = :NEW.material_id;
 11
 12    IF vEstatus_id = 'ES1' THEN
 13      UPDATE ejemplar SET estatus_id = 'ES2'
 14      WHERE material_id = :NEW.material_id
 15      AND noEjemplar = :NEW.noEjemplar;
 16      DBMS_OUTPUT.PUT_LINE('Se cambio el estatus a prestado del ejemplar: ' ||  :NEW.noEjemplar);
 17
 18    ELSIF vEstatus_id = 'ES2' THEN
 19      RAISE_APPLICATION_ERROR(-20096, 'El ejemplar '|| :NEW.noEjemplar || ' del material ' ||
:NEW.material_id || 'ya se encuentra prestado.');
 20
 21    ELSIF vEstatus_id = 'ES3' THEN
 22      RAISE_APPLICATION_ERROR(-20095, 'El ejemplar '|| :NEW.noEjemplar || ' del material ' ||
:NEW.material_id || 'no sale de la biblioteca.');
 23
 24    ELSE
```

```
 25      RAISE_APPLICATION_ERROR(-20094, 'El ejemplar '|| :NEW.noEjemplar || ' del material ' ||
:NEW.material_id || 'est  en mantenimiento.');
 26    END IF;
 27  END tgPrestamoEjemplar;
 28  /

Trigger created.

SQL>
SQL> --Pruebas para ejecutar
SQL> --INSERT INTO prestamo VALUES ('P'||SeqAltaPrestamo.NEXTVAL, 0, '08/10/16', SYSDATE, '10/10/16', 'L1',
'EJ1', 'M1');
SQL> --SELECT * FROM ejemplar e JOIN estatus s ON e.estatus_id=s.estatus_id;
SQL>
SQL> -----------3.- CHAVIRA        El resello de un material se realiza £nicamente en la fecha de vencimiento
del pr,stamo en funci¢n del tipo de lector. -- Chavira
SQL> CREATE OR REPLACE TRIGGER tgRevisarResello
  2  BEFORE UPDATE ON prestamo
  3  FOR EACH ROW
  4  BEGIN
  5    IF TRUNC(:OLD.fechaResello-SYSDATE)=0 THEN
  6      DBMS_OUTPUT.PUT_LINE('La fecha de resello coincide con la fecha actual.');
  7    ELSIF :OLD.fechaResello<SYSDATE THEN
  8      RAISE_APPLICATION_ERROR(-20097,'Este libro ya pas¢ su fecha de resello. Debe expedirse una multa.');
  9    ELSE
 10      RAISE_APPLICATION_ERROR(-20098, 'Este libro s¢lo se puede resellar el d¡a de su fecha de resello.');
 11    END IF;
 12  END tgRevisarResello;
 13  /

Trigger created.

SQL> ALTER TRIGGER tgRevisarResello DISABLE;

Trigger altered.

SQL>
SQL> -----------4.- JOYA           Al realizarse una devoluci¢n en tiempo, se eliminar  el pr,stamo.
SQL> CREATE OR REPLACE TRIGGER tgDevolEliminPrest
  2    AFTER DELETE
  3    ON prestamo
  4    FOR EACH ROW
  5    DECLARE
  6      vprestamo_id CHAR(5);
  7      vfechaVenci DATE;
  8    BEGIN
  9      IF :old.fechaVencimiento >= SYSDATE THEN
 10        DBMS_OUTPUT.PUT_LINE('Se elimin¢ prestamo con id ' ||  :old.prestamo_id);
 11      ELSE
 12        INSERT INTO multa
 13        VALUES('MU' || seqAltaMulta.NEXTVAL, :old.lector_id, SYSDATE, (TRUNC(SYSDATE -
:old.fechaVencimiento))*10, TRUNC(SYSDATE - :old.fechaVencimiento));
 14      END IF;
 15      UPDATE ejemplar SET estatus_id='ES1'
 16      WHERE noEjemplar=:old.noEjemplar AND material_id = :old.material_id;
 17    END tgDevolEliminPrest;
 18    /

Trigger created.

SQL>
SQL> spool off;
```

# DML

```
SQL> --Autor
SQL> exec AltaAutor('Catlaina', 'Gerritsma', 'Zipsell', 'Pakistani');
Se di¢ de alta al autor A0

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Andeee', 'Tuer', 'McGoldrick', 'Thai');
Se di¢ de alta al autor A1

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Annette', 'Paxforde', 'Ecob', 'Laotian');
Se di¢ de alta al autor A2

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Hyacintha', 'Bradtke', 'Pontin', 'Colombian');
Se di¢ de alta al autor A3

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Barbi', 'Spracklin', 'Kitt', 'Dominican');
Se di¢ de alta al autor A4

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Vernice', 'Dainter', 'Dryburgh', 'Panamanian');
Se di¢ de alta al autor A5

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Connie', 'Thring', 'Anderer', 'Alaska Native');
Se di¢ de alta al autor A6

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Brande', 'Dawson', 'Erni', 'Uruguayan');
Se di¢ de alta al autor A7

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Billi', 'Corking', 'Jellett', 'Fijian');
Se di¢ de alta al autor A8

PL/SQL procedure successfully completed.

SQL> exec AltaAutor('Riva', 'Beddingham', 'Dunleavy', 'Eskimo');
Se di¢ de alta al autor A9

PL/SQL procedure successfully completed.

SQL>
SQL> --Libros
SQL> declare
  2    vAutor_id char(10);
  3  begin
  4  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
  5  AltaLibro('sotano', 'arriba', 'Under the Domim Tree (Etz Hadomim Tafus)','L', vAutor_id, '651665651-7',
'n/a', '1');
```

```
  6  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
  7  AltaLibro('primer piso', 'abajo', 'Ricky Gervais: Out of England - The Stand-Up Special','L',vAutor_id,
'942371083-2', 'Public Utilities', '2');
  8  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
  9  AltaLibro('sotano', 'abajo', 'Beat the Devil','L', vAutor_id, '774114970-9', 'n/a', '3');
 10  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 11  AltaLibro('sotano', 'arriba', 'Rebellion (L''ordre et la morale)','L',vAutor_id, '467907549-X', 'n/a',
'4');
 12  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 13  AltaLibro('sotano', 'arriba', 'Uncertainty','L',vAutor_id, '867356448-4', 'n/a', '5');
 14  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 15  AltaLibro('primer piso', 'abajo', 'Letter from an Unknown Woman','L',vAutor_id, '570795478-3', 'n/a',
'6');
 16  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 17  AltaLibro('primer piso', 'arriba', 'Wagner''s Dream','L',vAutor_id, '256681977-7', 'Consumer Services',
'7');
 18  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 19  AltaLibro('segundo piso', 'abajo', 'Thick-Walled Room, The (Kabe atsuki heya)','L',vAutor_id,
'343065905-1', 'n/a', '8');
 20  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 21  AltaLibro('sotano', 'abajo', 'SS Camp 5: Women''s Hell (SS Lager 5: L''inferno delle donne)','L',
vAutor_id, '272771509-9', 'n/a', '9');
 22  select  autor_id into vAutor_id from  (select autor_id from autor  order by dbms_random.value)where
rownum=1;
 23  AltaLibro('segundo piso', 'abajo', 'Kid from Brooklyn, The','L', vAutor_id, '385322086-X', 'Consumer
Services', '10');
 24
 25  end;
 26  /
Se dio de alta exitosamente al libro M0
Se dio de alta exitosamente al libro M1
Se dio de alta exitosamente al libro M2
Se dio de alta exitosamente al libro M3
Se dio de alta exitosamente al libro M4
Se dio de alta exitosamente al libro M5
Se dio de alta exitosamente al libro M6
Se dio de alta exitosamente al libro M7
Se dio de alta exitosamente al libro M8
Se dio de alta exitosamente al libro M9

PL/SQL procedure successfully completed.

SQL>
SQL> --grado academico
SQL> INSERT INTO gradoAcademico VALUES ('GA1', 'Universidad');

1 row created.

SQL> INSERT INTO gradoAcademico VALUES ('GA2', 'Maestria');

1 row created.

SQL> INSERT INTO gradoAcademico VALUES ('GA3', 'Doctorado');

1 row created.
```

```
SQL>
SQL>
SQL> --director tesis
SQL> declare
  2    vDescripcionGA varchar2(20);
  3  begin
  4  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
  5  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'G,raldine', 'Meriot', 'Coulthard');
  6  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
  7  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'Marie-fran‡oise', 'Refford', 'Kelleway');
  8  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
  9  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'Personnalis,e', 'Calton', 'Bogart');
 10  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 11  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'DaniŠle', 'Klimashevich', 'Tarbin');
 12  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 13  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'Eug,nie', 'Maddra', 'Challiss');
 14  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 15  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'Est,e', 'Rounsefell', 'Arthey');
 16  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 17  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'AdŠle', 'Fairchild', 'McSperrin');
 18  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 19  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'YŠ', 'Guilloton', 'Riddington');
 20  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 21  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'Ma‹lys', 'Fussen', 'Binnall');
 22  select descripcionGA into vDescripcionGA from (select * from gradoacademico order by dbms_random.value)
where rownum=1;
 23  AltaDirTesis(SeqAltaDirectorTesis.nextval, vDescripcionGA, 'Cl,mentine', 'Tole', 'Gaffon');
 24  end;
 25  /
Se insert¢ al director de tesis con id: D0
Se insert¢ al director de tesis con id: D1
Se insert¢ al director de tesis con id: D2
Se insert¢ al director de tesis con id: D3
Se insert¢ al director de tesis con id: D4
Se insert¢ al director de tesis con id: D5
Se insert¢ al director de tesis con id: D6
Se insert¢ al director de tesis con id: D7
Se insert¢ al director de tesis con id: D8
Se insert¢ al director de tesis con id: D9

PL/SQL procedure successfully completed.

SQL>
SQL> --Tesis
SQL> EXEC AltaTesis('Ingenieria','2000','D0','Mexico','Abajo','Algebra1','A0');
Se inserto un nuevo material tipo tesis:  M10

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Aquitectura','2010','D2','EU','Arriba','Recursos de Construccion','A1');
Se inserto un nuevo material tipo tesis:  M11

PL/SQL procedure successfully completed.
```

```
SQL> EXEC AltaTesis('Bilogia','2002','D1','Brasil','Arriba','Biologia Celular','A2');
Se inserto un nuevo material tipo tesis:  M12

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Fisica','1993','D2','Canada','Abajo','Newton','A3');
Se inserto un nuevo material tipo tesis:  M13

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Fisica','2003','D3','Mexico','AbaArribajo','Caida Libre','A3');
Se inserto un nuevo material tipo tesis:  M14

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Ingenieria','2004','D6','Reino Unido','Arriba','Calculo V','A4');
Se inserto un nuevo material tipo tesis:  M15

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Electricidad','2009','D6','Venezuela','Abajo','Ley de Ohm','A7');
Se inserto un nuevo material tipo tesis:  M16

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Contabilidad','2019','D8','Colombia','Arriba','Sumas y restas','A4');
Se inserto un nuevo material tipo tesis:  M17

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Medicina','2001','D9','Mexico','Abajo','Medicina Fam','A5');
Se inserto un nuevo material tipo tesis:  M18

PL/SQL procedure successfully completed.

SQL> EXEC AltaTesis('Veterinaria','2000','D4','Peru','Arriba','Perritos 1','A3');
Se inserto un nuevo material tipo tesis:  M19

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL> --estatus
SQL> INSERT INTO estatus VALUES ('ES1', 'Disponible');

1 row created.

SQL> INSERT INTO estatus VALUES ('ES2', 'Prestado');

1 row created.

SQL> INSERT INTO estatus VALUES ('ES3', 'No deja biblioteca');

1 row created.

SQL> INSERT INTO estatus VALUES ('ES4', 'En mantenimiento');

1 row created.

SQL>
SQL> --ejemplares
SQL> EXEC AltaEjemplar('M1');
```

```
PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M1');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M1');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M2');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M2');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M12');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M13');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M14');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M15');

PL/SQL procedure successfully completed.

SQL> EXEC AltaEjemplar('M16');

PL/SQL procedure successfully completed.

SQL>
SQL>
SQL>
SQL> --tipo lector
SQL> INSERT INTO tipoLector VALUES ('TL1', 'Estudiante', '3', '1','8' );

1 row created.

SQL> INSERT INTO tipoLector VALUES ('TL2', 'Profesor', '5', '2','15' );

1 row created.

SQL> INSERT INTO tipoLector VALUES ('TL3', 'Investigador', '10', '3','30' );

1 row created.

SQL>
SQL> --lectores
SQL> declare
  2    vTipoLector char(3);
  3  begin
  4  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
```

```
  5  AltaLector('0983813282', 'Carey', 'Rustan', 'Ansell', 'CDMX', '4', 'calle-75', 'colonia-00',
'delegacion-48', vTipoLector);
  6  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
  7  AltaLector('1953180047', 'Casey', 'Sapsforde', 'Wroe', 'CDMX', '5', 'calle-50', 'colonia-72',
'delegacion-00', vTipoLector);
  8  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
  9  AltaLector('6886594908', 'Alisun', 'Eckery', 'Linner', 'BC', '4', 'calle-89', 'colonia-52', 'delegacion-
20', vTipoLector);
 10  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 11  AltaLector('5707838082', 'Krissie', 'Houndson', 'Lafee', 'CDMX', '4', 'calle-94', 'colonia-46',
'delegacion-55', vTipoLector);
 12  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 13  AltaLector('4568285599', 'Danit', 'Yitzhakov', 'Letessier', 'BC', '7', 'calle-46', 'colonia-56',
'delegacion-20', vTipoLector);
 14  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 15  AltaLector('7112882060', 'Mychal', 'Wooffinden', 'Barkworth', 'BC', '7', 'calle-76', 'colonia-33',
'delegacion-10', vTipoLector);
 16  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 17  AltaLector('0028418167', 'Neysa', 'Bytheway', 'Readwing', 'CDMX', '7', 'calle-42', 'colonia-70',
'delegacion-13', vTipoLector);
 18  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 19  AltaLector('4509672820', 'Erna', 'Jozwicki', 'Duerdin', 'CDMX', '0', 'calle-70', 'colonia-52',
'delegacion-42', vTipoLector);
 20  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 21  AltaLector('3759962312', 'Tynan', 'Murrey', 'Philipsen', 'Puebla', '2', 'calle-30', 'colonia-75',
'delegacion-33', vTipoLector);
 22  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 23  AltaLector('8702003894', 'Sacha', 'Scading', 'McLagan', 'Puebla', '4', 'calle-52', 'colonia-22',
'delegacion-33', vTipoLector);
 24  select tipolector_id into vTipoLector from (select tipolector_id from tipolector order by
dbms_random.value) where rownum=1;
 25
 26  end;
 27  /
Alta de Usuario L1         exitosa
Alta de Usuario L2         exitosa
Alta de Usuario L3         exitosa
Alta de Usuario L4         exitosa
Alta de Usuario L5         exitosa
Alta de Usuario L6         exitosa
Alta de Usuario L7         exitosa
Alta de Usuario L8         exitosa
Alta de Usuario L9         exitosa
Alta de Usuario L10        exitosa

PL/SQL procedure successfully completed.

SQL>
SQL> --prestamos
SQL> EXEC altaprestamo('L1', 'EJ0', 'M1');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M1 Al lector: L1

PL/SQL procedure successfully completed.
```

```
SQL> EXEC altaprestamo('L1', 'EJ0', 'M2');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M2 Al lector: L1

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L1', 'EJ1', 'M1');
Se cambio el estatus a prestado del ejemplar: EJ1
Se realiz¢ el prestamo del material:  M1 Al lector: L1

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L3', 'EJ0', 'M12');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M12 Al lector: L3

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L4', 'EJ2', 'M1');
Se cambio el estatus a prestado del ejemplar: EJ2
Se realiz¢ el prestamo del material:  M1 Al lector: L4

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L5', 'EJ1', 'M2');
Se cambio el estatus a prestado del ejemplar: EJ1
Se realiz¢ el prestamo del material:  M2 Al lector: L5

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L6', 'EJ0', 'M13');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M13 Al lector: L6

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L7', 'EJ0', 'M14');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M14 Al lector: L7

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L8', 'EJ0', 'M16');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M16 Al lector: L8

PL/SQL procedure successfully completed.

SQL> EXEC altaprestamo('L9', 'EJ0', 'M15');
Se cambio el estatus a prestado del ejemplar: EJ0
Se realiz¢ el prestamo del material:  M15 Al lector: L9

PL/SQL procedure successfully completed.

SQL>
SQL> UPDATE prestamo SET fecharesello = SYSDATE - 20, fechaprestamo = SYSDATE - 20, fechavencimiento =
SYSDATE - 12 WHERE LECTOR_ID = 'L5';

1 row updated.

SQL> UPDATE prestamo SET fecharesello = SYSDATE - 40, fechaprestamo = SYSDATE - 40, fechavencimiento =
SYSDATE - 20 WHERE LECTOR_ID = 'L6';
```

```
1 row updated.

SQL> UPDATE prestamo SET fecharesello = SYSDATE - 30, fechaprestamo = SYSDATE - 30, fechavencimiento =
SYSDATE - 12 WHERE LECTOR_ID = 'L7';

1 row updated.

SQL> UPDATE prestamo SET fecharesello = SYSDATE - 26, fechaprestamo = SYSDATE - 26, fechavencimiento =
SYSDATE - 5 WHERE LECTOR_ID = 'L8';

1 row updated.

SQL> UPDATE prestamo SET fecharesello = SYSDATE - 20, fechaprestamo = SYSDATE - 20, fechavencimiento =
SYSDATE - 12 WHERE LECTOR_ID = 'L9';

1 row updated.

SQL> --SE TIENE QUE EJECUTAR TRIGGER tgDevolEliminPrest ANTES DE LO SIGUIENTE
SQL>
SQL> --Multas
SQL> DELETE prestamo WHERE prestamo_id = 'P0';
Se elimin¢ prestamo con id P0

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P1';
Se elimin¢ prestamo con id P1

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P2';
Se elimin¢ prestamo con id P2

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P3';
Se elimin¢ prestamo con id P3

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P4';
Se elimin¢ prestamo con id P4

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P5';

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P6';

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P7';

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P8';

1 row deleted.

SQL> DELETE prestamo WHERE prestamo_id = 'P9';
```

```
1 row deleted.

SQL>
SQL> ALTER TRIGGER tgRevisarResello ENABLE;

Trigger altered.

SQL>
SQL> spool off;
**********************************************************************************************************
SQL> --------------------1.- OSCAR       Vista de los cat logos de LIBRO.
SQL> CREATE OR REPLACE VIEW vwCatalogoLibro(
  2    material_id,
  3    titulo,
  4    autor,
  5    tema,
  6    edicion
  7  )
  8  AS
  9  SELECT m.material_id, m.titulo, a.autor_id, l.tema, l.edicion
 10  FROM material m
 11  JOIN libro l   ON l.material_id = m.material_id
 12  JOIN escribe e  ON e.material_id = m.material_id
 13  JOIN autor a    ON a.autor_id = e.autor_id;

View created.

SQL>
SQL> --------------------2.- CHAVIRA     Vista de los cat logos de TESIS.
SQL> CREATE OR REPLACE VIEW vwCatalogoTesis
  2  AS
  3  SELECT t.material_id, t.tesis_id, m.titulo, a.autor_id, (a.nombreAutor||' '||a.apPaternoAutor)
NombreAutor, t.anoPublicacion, t.director_id
  4  FROM tesis t
  5  JOIN escribe e ON t.material_id=e.material_id
  6  JOIN autor a ON e.autor_id=a.autor_id
  7  JOIN material m ON t.material_id=m.material_id
  8  JOIN ejemplar On t.material_id=ejemplar.material_id
  9  JOIN directorTesis d ON t.director_id=d.director_id;

View created.

SQL>
SQL> --------------------3.- JOYA        Vista de los cat logos de AUTOR con su tipo de material.
SQL> CREATE OR REPLACE VIEW vwCatalogoAutor(
  2    AutorId, NombreAutor, ApPaterno, Titulo, TipoMaterial
  3  )
  4  AS
  5  SELECT e.autor_id, a.nombreAutor, a.apPaternoAutor, m.titulo, m.tipoMaterial
  6  FROM autor a
  7  JOIN escribe e ON (e.autor_id = a.autor_id)
  8  JOIN material m ON (e.material_id = m.material_id);

View created.

SQL>
SQL> --------------------4.- LAZARO      Vista de estatus del MATERIAL.
SQL> create or replace view vwCatalogoEstatusMaterial as
  2  select titulo,noEjemplar,descripcione
  3  from material
  4  join ejemplar using(material_id)
  5  join estatus using(estatus_id);
```

# Vistas

```
View created.

SQL>
SQL> --------------------5.- OSCAR       Vista de LECTOR por tipo.
SQL>
SQL> CREATE OR REPLACE VIEW vwCatalogoLector(
  2    lector_id,
  3    nombreLector,
  4    apPaternoLector,
  5    descripcionTipoLector
  6  )
  7  AS
  8  SELECT l.lector_id, l.nombreLector, l.apPaternoLector, tl.descripcionTipoLector
  9  FROM lector l
 10  JOIN tipoLector tl
 11  ON tl.tipoLector_id = l.tipoLector_id
 12  ORDER BY tl.descripcionTipoLector;

View created.

SQL> spool off;
```