Solución Ejercicio Prac. 3b (Esquema).

LIBRO(codlib, título, autores, temática, totalpres)

CP= {codlib}

VNN= {título}

PRÉSTAMO(codlib, codsoc, fechapres, fechadev)

CP= {codlib, fechapres}

VNN= {codsoc}

CAj= {codlib} $\rightarrow$ LIBRO

CAj= {codsoc} $\rightarrow$ SOCIO

SOCIO(codsoc, nombre, dir, tel)

CP= {codsoc}

VNN= {nombre}

Solución Ejercicio Prac. 3b (Esquema).

R1:

LX:LIBRO

$\forall LX\ ((LIBRO(LX) \wedge (\neg nulo(LX.temática))$

$\rightarrow (\quad (LX.temática = \text{‘Física’}) \vee$

$(LX.temática = \text{‘Óptica’}\ ) \vee$

$(LX.temática = \text{‘Mecánica’}\ ) \vee$

$(LX.temática = \text{‘Electricidad’}) \quad))$

R2:

PX:PRÉSTAMO

$\forall PX\ ((PRÉSTAMO(PX) \wedge (\neg nulo(PX.fechadev))$

$\rightarrow (PX.fechapres \leq PX.fechadev))$

Faltarían algunas restricciones, como que no haya dos préstamos del mismo libro con fechas solapadas.

Solución Parcial Ejercicio (SQL).

```
CREATE TABLE socio
    ( codsoc NUMBER constraint cp_socio PRIMARY KEY deferrable,
      nombre varchar(40) NOT NULL,          dir varchar(60),
      tel varchar(15),                      totalpres number default 0 not null deferrable );
CREATE TABLE libro
    (codlib number constraint cp_libro PRIMARY KEY deferrable,
     titulo varchar(50) NOT NULL,           autores varchar(60) NOT NULL,
    tematica varchar(20),
    constraint r1 CHECK (tematica IN ('Fisica','Optica','Mecanica','Electricidad')) );
CREATE TABLE prestamo
    (codlib number, codsoc number NOT NULL, fechapres date, fechadev date,
      constraint cp_prestamo PRIMARY KEY(codlib, fechapres) deferrable,
      constraint ca_pre_libro FOREIGN KEY (codlib) REFERENCES libro  deferrable,
      constraint ca_pre_socio FOREIGN KEY (codsoc) REFERENCES socio  deferrable,
     constraint R2 CHECK (fechapres <= fechadev)  deferrable    );
```

Solución Ejercicio (SQL).

Queda mantener el dato “totalpres”...

Solución ORACLE:
TRIGGERS

Solución Ejercicio (SQL).

EVENTOS que deben afectar a “totalpres”:

	EVENTO			CONDICIÓN	ACCIÓN
	Operación	Tabla	Atributo		
Imprescindible	INSERT	PRESTAMO	-	nula(fechadev)	+1
	UPDATE	PRESTAMO	fechadev	de nula a no nula	-1
				de no nula a nula	+1 (o no perm.)
	UPDATE	PRESTAMO	codsoc	nula(fechadev)	+1 y -1
	DELETE	PRESTAMO	-	nula(fechadev)	-1 (o no perm.)
	INSERT	SOCIO	-	si <> 0	no permitir
	UPDATE	SOCIO	totalpres	si no correcto	no permitir

Solución Ejercicio (SQL).

Se ha de añadir un valor por defecto a totalpres en la definición de la tabla “socio” de la siguiente manera:

totalpres DEFAULT 0

Y se deben crear los siguientes triggers para mantener el atributo derivado “totalpres”.

Solución Ejercicio (SQL).

El siguiente disparador controla la inserción de nuevos préstamos:

```
CREATE TRIGGER T1
AFTER INSERT ON Prestamo
FOR EACH ROW
BEGIN
    IF :new.fechadev IS NULL THEN
        UPDATE socio SET totalpres = totalpres +1
                        WHERE codsoc = :new.codsoc;
    END IF;
END;
```

Solución Ejercicio (SQL).

El siguiente disparador controla la modificación del atributo “fechadev” de préstamos (se devuelve o se modifica a prestado un libro):

```
CREATE TRIGGER T2
AFTER UPDATE OF fechadev ON Prestamo
FOR EACH ROW
BEGIN
    IF :new.fechadev IS NULL AND :old.fechadev IS NOT NULL THEN
        UPDATE socio SET totalpres = totalpres +1 WHERE codsoc = :new.codsoc;
    ELSE IF :new.fechadev IS NOT NULL AND :old.fechadev IS NULL THEN
        UPDATE socio SET totalpres = totalpres -1  WHERE codsoc = :new.codsoc;
        END IF;
    END IF;
END;
```

Solución Ejercicio (SQL).

El siguiente disparador controla la modificación del atributo “codsoc” de préstamos (se le asigna un préstamo a otro socio):

```
CREATE TRIGGER T3
AFTER UPDATE OF codsoc ON Prestamo
FOR EACH ROW
BEGIN
    IF :new.fechadev IS NULL THEN
        UPDATE socio SET totalpres = totalpres +1 WHERE codsoc = :new.codsoc;
        UPDATE socio SET totalpres = totalpres -1  WHERE codsoc = :old.codsoc;
    END IF;
END;
```

Solución Ejercicio (SQL).

El siguiente disparador controla el borrado de préstamos (sólo se puede hacer si el libro se ha devuelto):

```
CREATE TRIGGER T4
AFTER DELETE ON Prestamo
FOR EACH ROW
BEGIN
    IF :old.fechadev IS NULL THEN
        UPDATE socio SET totalpres = totalpres -1
                                WHERE codsoc = :old.codsoc;
    END IF;
END;
```

Solución Ejercicio (SQL).

El siguiente disparador controla que el número de préstamos siempre sea 0 al insertar un nuevo socio:

```
CREATE TRIGGER T5
AFTER INSERT ON Socio
FOR EACH ROW
BEGIN
  IF :new.totalpres <> 0 then
        RAISE_APPLICATION_ERROR(-20000, 'Cuando se inserta un socio,
                                su número de préstamos ha de ser 0.');
  END IF;
END;
```

Solución Ejercicio (SQL).

El siguiente disparador controla que el número de préstamos de un socio no se pueda modificar:

```
CREATE TRIGGER T6
AFTER UPDATE OF totalpres ON Socio
FOR EACH ROW
DECLARE
  N NUMBER;
BEGIN
  SELECT COUNT(*) INTO N
  FROM PRESTAMO P
  WHERE P.fechadev IS NULL AND P.codsoc = :new.codsoc;
   IF N <> :new.totalpres THEN
      RAISE_APPLICATION_ERROR(-20000, 'Modificación no consistente.');
  END IF;
END;
```

Este trigger podría dar problemas porque puede llamarse por otros triggers (T1-T4).

Una opción mejor es no permitir la modificación de totalpres a ningún usuario utilizando permisos:

REVOKE UPATE(totalpres) TO ALL;