**Reportes**

**--R.1. Reporte de libros m�s populares: Muestra una lista de los libros m�s prestados o solicitados por los usuarios en un per�odo de tiempo espec�fico.**

SELECT

MB.titulo\_mat\_biblio AS "Título del Libro",

COUNT(\*) AS "Cantidad de Préstamos"

FROM

Prestamo P

INNER JOIN Material\_bibliografico MB ON P.id\_mat\_biblio = MB.id\_mat\_biblio

WHERE

P.fecha\_prestamo BETWEEN TO\_DATE('2023-01-01', 'YYYY-MM-DD') AND TO\_DATE('2023-06-01', 'YYYY-MM-DD') -- Período de tiempo específico

GROUP BY

MB.titulo\_mat\_biblio

ORDER BY

COUNT(\*) DESC;

**--R.2. Reporte de pr�stamos vencidos: Muestra una lista de los pr�stamos que est�n vencidos y a�n no han sido devueltos por los usuarios, incluyendo detalles como el nombre del usuario y la fecha de vencimiento.**

SELECT

U.nombreU AS "Nombre del Usuario",

P.fecha\_devolucion AS "Fecha de Vencimiento",

MB.titulo\_mat\_biblio AS "Nombre del Libro"

FROM

Prestamo P

INNER JOIN Usuario U ON P.id\_usuario = U.id\_usuario

INNER JOIN Material\_bibliografico MB ON P.id\_mat\_biblio = MB.id\_mat\_biblio

WHERE

P.fecha\_devolucion < SYSDATE -- Préstamos vencidos

AND P.estado\_prestamo != 'devuelto' -- Préstamos no devueltos

ORDER BY

P.fecha\_devolucion ASC;

**---R.3. Reporte de libros en reserva: Muestra una lista de los libros que han sido reservados por los usuarios y su estado actual (disponibles, en pr�stamo o vencidos).**

SELECT

MB.titulo\_mat\_biblio AS "Nombre del Libro",

CASE

WHEN P.estado\_prestamo = 'reservado' THEN 'En reserva'

WHEN P.estado\_prestamo = 'prestado' THEN 'En préstamo'

WHEN P.fecha\_devolucion < SYSDATE THEN 'Vencido'

ELSE 'Disponible'

END AS "Estado"

FROM

Material\_bibliografico MB

INNER JOIN Prestamo P ON MB.id\_mat\_biblio = P.id\_mat\_biblio

WHERE

P.estado\_prestamo = 'reservado' -- Solo libros en reserva

OR P.estado\_prestamo = 'prestado' -- También incluye libros en préstamo

OR P.fecha\_devolucion < SYSDATE -- También incluye libros vencidos

ORDER BY

MB.titulo\_mat\_biblio ASC;

**---R.4. Reporte de usuarios con multas pendientes: Muestra una lista de los usuarios que tienen multas pendientes por pr�stamos vencidos o incumplimientos de las pol�ticas de la biblioteca, junto con el monto total de las multas.**

SELECT

U.nombreU AS "Nombre del Usuario",

M.id\_multa AS "ID de la Multa",

M.fecha\_imposicion AS "Fecha de la Multa",

M.monto\_multa AS "Monto de la Multa Bs."

FROM

Usuario U

INNER JOIN Multa M ON U.id\_usuario = M.id\_usuario

WHERE

M.estado = 'pendiente' -- Multas pendientes

ORDER BY

U.nombreU ASC;

**--- R.5 Reporte de estad�sticas de uso por categor�a: Muestra estad�sticas sobre el uso de libros por categor�a, como el n�mero de pr�stamos, la duraci�n promedio de los pr�stamos y los libros m�s populares en cada categor�a.**

SELECT

C.categoria AS "Categoría",

COUNT(P.id\_prestamo) AS "Núm. Préstamos",

AVG(EXTRACT(DAY FROM (P.fecha\_devolucion - P.fecha\_prestamo))) AS "Duración Prom. (Días)",

MB.titulo\_mat\_biblio AS "Libro más Popular"

FROM

Categoria C

INNER JOIN tiene\_categoria TC ON C.id\_categoria = TC.id\_categoria

INNER JOIN Material\_bibliografico MB ON TC.id\_mat\_biblio = MB.id\_mat\_biblio

LEFT JOIN Prestamo P ON MB.id\_mat\_biblio = P.id\_mat\_biblio

GROUP BY

C.categoria, MB.titulo\_mat\_biblio

ORDER BY

C.categoria ASC, COUNT(P.id\_prestamo) DESC;

**---Reporte de adquisiciones recientes: Muestra una lista de las adquisiciones m�s recientes realizadas por la biblioteca, incluyendo detalles como el t�tulo del libro, el autor, la fecha de adquisici�n y el costo.**

SELECT

MB.titulo\_mat\_biblio AS "Título del Libro",

A.autores AS "Autores",

AD.fecha\_adquisicion AS "Fecha de Adquisición",

AD.presupuesto AS "Costo",

AD.tasacion AS "tasacion"

FROM

Material\_bibliografico MB

INNER JOIN Ejemplar E ON MB.id\_mat\_biblio = E.id\_mat\_biblio

INNER JOIN adquisicion AD ON E.id\_adquisicion = AD.id\_adquisicion

INNER JOIN (

SELECT

EP.id\_mat\_biblio,

LISTAGG(A.nombre, ', ') WITHIN GROUP (ORDER BY A.nombre) AS autores

FROM

escrito\_por EP

INNER JOIN Autor A ON EP.id\_autor = A.id\_autor

GROUP BY

EP.id\_mat\_biblio

) A ON MB.id\_mat\_biblio = A.id\_mat\_biblio

ORDER BY

AD.fecha\_adquisicion DESC;

**---R.7. Reporte de usuarios activos: Muestra una lista de los usuarios activos en la biblioteca, incluyendo detalles como su nombre, n�mero de identificaci�n, fecha de registro y estado de la membres�a.**

SELECT

U.nombreU AS "Nombre del Usuario",

U.ci AS "Número de Identificación",

U.fecha\_registro AS "Fecha de Registro",

L.categoria\_membresia AS "Estado de Membresía"

FROM

Usuario U

INNER JOIN Lector L ON U.id\_usuario = L.id\_usuario

WHERE

L.fecha\_vencimiento\_membresia >= SYSDATE

ORDER BY

U.nombreU ASC;

**---R.8. Reporte de libros por sede: Muestra una lista de los libros disponibles en cada sede de la biblioteca, junto con la cantidad de ejemplares disponibles en cada sede.**

SELECT

S.nombre\_sede AS "Sede",

MB.titulo\_mat\_biblio AS "Título del Libro",

COUNT(E.cod\_serial) AS "Cantidad de Ejemplares"

FROM

Sede S

INNER JOIN Ejemplar E ON S.id\_sede = E.id\_sede

INNER JOIN Material\_bibliografico MB ON E.id\_mat\_biblio = MB.id\_mat\_biblio

WHERE

E.cod\_serial NOT IN (

SELECT P.cod\_serial FROM Prestamo P WHERE P.estado\_prestamo = 'prestado'

)

GROUP BY

S.nombre\_sede, MB.titulo\_mat\_biblio

ORDER BY

S.nombre\_sede ASC, MB.titulo\_mat\_biblio ASC;

**---R.9. Reporte de libros por estanter�a virtual: Muestra una lista de los libros organizados en las estanter�as virtuales, categorizados por estanter�as privadas, p�blicas y libres, junto con la informaci�n del propietario de cada estanter�a.**

SELECT

E.nombre\_estanteria\_virtual AS "Estantería Virtual",

CASE

WHEN E.tipo\_estanteria\_virtual = 'privada' THEN 'Privada'

WHEN E.tipo\_estanteria\_virtual = 'publica' THEN 'Pública'

ELSE 'Libre'

END AS "Tipo de Estantería",

U.nombreU AS "Propietario de la Estantería",

MB.titulo\_mat\_biblio AS "Título del Libro"

FROM

Estanteria\_virtual E

LEFT JOIN Usuario U ON E.id\_usuario = U.id\_usuario

LEFT JOIN Adiciona\_estanteria AE ON E.id\_estanteria\_virtual = AE.id\_estanteria\_virtual

LEFT JOIN Material\_bibliografico MB ON AE.id\_mat\_biblio = MB.id\_mat\_biblio

ORDER BY

"Tipo de Estantería" ASC, E.nombre\_estanteria\_virtual ASC, MB.titulo\_mat\_biblio ASC;

**---R.10. Reporte de pr�stamos por per�odo de tiempo: Muestra estad�sticas sobre los pr�stamos realizados en un per�odo de tiempo espec�fico, incluyendo el n�mero total de pr�stamos, la duraci�n promedio de los pr�stamos y la tasa de devoluci�n puntual.**

SELECT

COUNT(id\_prestamo) AS "Número total de Préstamos",

AVG(EXTRACT(DAY FROM fecha\_devolucion - fecha\_prestamo)) AS "Duración (Días) de Préstamo"

FROM

Prestamo

WHERE

fecha\_prestamo >= TO\_DATE('2023-01-01', 'YYYY-MM-DD')

AND fecha\_prestamo <= TO\_DATE('2023-12-31', 'YYYY-MM-DD');

**Procedimientos**

**PA1. SP\_BusquedaLibros: Este procedimiento acepta parámetros de búsqueda y devuelve una lista de libros que coinciden con los criterios de búsqueda configurados.**

CREATE OR REPLACE PROCEDURE SP\_BusquedaLibros (

p\_Titulo IN VARCHAR2 DEFAULT NULL,

p\_Autor IN VARCHAR2 DEFAULT NULL,

p\_Editorial IN VARCHAR2 DEFAULT NULL

)

IS

v\_Titulo VARCHAR2(100);

v\_Autores VARCHAR2(100);

v\_Editorial VARCHAR2(100);

BEGIN

SELECT

MB.titulo\_material\_bibliografico AS "Título del Libro",

A.autores AS "Autores",

E.nombre\_editorial AS "Editorial"

INTO

v\_Titulo,

v\_Autores,

v\_Editorial

FROM

Material\_bibliografico MB

LEFT JOIN (

SELECT

EP.id\_material\_biblio,

LISTAGG(A."nombre(s)", ', ') WITHIN GROUP (ORDER BY A."nombre(s)") AS autores

FROM

escrito\_por EP

INNER JOIN Autor A ON EP.id\_autor = A.id\_autor

GROUP BY

EP.id\_material\_biblio

) A ON MB.id\_material\_biblio = A.id\_material\_biblio

LEFT JOIN Editorial E ON MB.id\_editorial = E.id\_editorial

WHERE

(p\_Titulo IS NULL OR MB.titulo\_material\_bibliografico LIKE '%' || p\_Titulo || '%')

AND (p\_Autor IS NULL OR A.autores LIKE '%' || p\_Autor || '%')

AND (p\_Editorial IS NULL OR E.nombre\_editorial LIKE '%' || p\_Editorial || '%');

-- Mostrar los resultados

DBMS\_OUTPUT.PUT\_LINE('Título del Libro: ' || v\_Titulo);

DBMS\_OUTPUT.PUT\_LINE('Autores: ' || v\_Autores);

DBMS\_OUTPUT.PUT\_LINE('Editorial: ' || v\_Editorial);

END;

**---PA2. SP\_ListadosLectura:**

**---Este procedimiento recibe como entrada el identificador del usuario y devuelve un listado de los libros que el usuario ha marcado como lectura.**

CREATE OR REPLACE PROCEDURE SP\_ListadosLectura (

p\_IdUsuario IN VARCHAR2

)

AS

BEGIN

SELECT

MB.titulo\_material\_bibliografico AS "Título del Libro",

LISTAGG(A.nombre, ', ') WITHIN GROUP (ORDER BY A.id\_autor) AS "Autores",

E.nombre\_editorial AS "Editorial"

FROM

Prestamo P

INNER JOIN Material\_bibliografico MB ON P.id\_material\_biblio = MB.id\_material\_biblio

LEFT JOIN Escrito\_por EP ON MB.id\_material\_biblio = EP.id\_material\_biblio

LEFT JOIN Autor A ON EP.id\_autor = A.id\_autor

LEFT JOIN Editorial E ON MB.id\_editorial = E.id\_editorial

WHERE

P.id\_usuario = IdUsuario AND P.estado\_prestamo = 'devuelto'

GROUP BY

MB.titulo\_material\_bibliografico,

E.nombre\_editorial;

END;

**PA3. SP\_AgregarAdquisicion: Este procedimiento permite agregar una nueva adquisición a la base de datos, incluyendo información de presupuesto e información de tasación.**

CREATE OR REPLACE PROCEDURE SP\_AgregarAdquisicion (

p\_id\_usuario IN VARCHAR2,

p\_id\_adquisicion IN VARCHAR2,

p\_id\_proveedor IN VARCHAR2,

p\_presupuesto IN NUMBER,

p\_tasacion IN NUMBER,

p\_fecha\_adquisicion IN DATE,

p\_cod\_serial IN VARCHAR2,

p\_id\_material\_biblio IN VARCHAR2,

p\_id\_editorial IN VARCHAR2,

p\_id\_Sede IN VARCHAR2,

p\_id\_estanteria IN VARCHAR2,

p\_id\_seccion IN VARCHAR2,

p\_id\_tipo IN NUMBER,

p\_numero\_paginas IN NUMBER,

p\_id\_idioma IN VARCHAR2

) AS

BEGIN

-- Verifica si la adquisición existe

IF EXISTS (

SELECT 1

FROM Adquisicion

WHERE id\_adquisicion = p\_id\_adquisicion

) THEN

SELECT 'La adquisición ya existe' AS Mensaje FROM dual;

RETURN;

END IF;

-- Insertamos la adquisición

INSERT INTO Adquisicion (id\_usuario, id\_adquisicion, id\_proveedor, presupuesto, tasacion, fecha\_adquisicion)

VALUES (p\_id\_usuario, p\_id\_adquisicion, p\_id\_proveedor, p\_presupuesto, p\_tasacion, p\_fecha\_adquisicion);

-- Insertamos los ejemplares

INSERT INTO Ejemplar (id\_usuario, cod\_serial, id\_material\_biblio, id\_editorial, id\_adquisicion, id\_proveedor, id\_Sede, id\_estanteria, id\_seccion, id\_tipo, numero\_paginas, id\_idioma)

VALUES (p\_id\_usuario, p\_cod\_serial, p\_id\_material\_biblio, p\_id\_editorial, p\_id\_adquisicion, p\_id\_proveedor, p\_id\_Sede, p\_id\_estanteria, p\_id\_seccion, p\_id\_tipo, p\_numero\_paginas, p\_id\_idioma);

-- Insertamos el material bibliográfico si no existe

IF NOT EXISTS (

SELECT 1

FROM Material\_bibliografico

WHERE id\_material\_biblio = p\_id\_material\_biblio

) THEN

INSERT INTO Material\_bibliografico (id\_material\_biblio, id\_editorial, id\_Sede, id\_estanteria, id\_seccion, id\_tipo, fecha\_publicacion, descripcion\_material\_bibliografico, titulo\_material\_bibliografico)

VALUES (p\_id\_material\_biblio, p\_id\_editorial, p\_id\_Sede, p\_id\_estanteria, p\_id\_seccion, p\_id\_tipo, SYSDATE, NULL, NULL);

END IF;

-- Mensaje de éxito

SELECT 'La adquisición se agregó correctamente' AS Mensaje FROM dual;

END;

**PA4. SP\_GestionSedes: Este procedimiento permite administrar las sedes de la biblioteca, incluyendo la capacidad de agregar, modificar o eliminar sedes.**

CREATE OR REPLACE PROCEDURE SP\_GestionSedes (

Accion IN VARCHAR2, -- Valores posibles: 'Agregar', 'Modificar', 'Eliminar'

Id\_Sede IN VARCHAR2,

NombreSede IN VARCHAR2,

PaisSede IN VARCHAR2,

CiudadSede IN VARCHAR2,

AvenidaCalle IN VARCHAR2,

NumeroSede IN NUMBER,

Capacidad IN NUMBER

) AS

BEGIN

IF Accion = 'Agregar' THEN

-- Verificar si la sede ya existe

SELECT COUNT(\*)

INTO Idsede

FROM Sede

Where id\_sede = IdSede;

If IdSede > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'La sede ya existe.');

END IF;

-- Insertar la nueva sede

INSERT INTO Sede (id\_Sede, nombre\_sede, pais\_Sede, ciudad\_Sede, avenida\_calle, numero\_sede, capacidad\_sede)

VALUES (IdSede, NombreSede, PaisSede, CiudadSede, AvenidaCalle, NumeroSede, Capacidad);

DBMS\_OUTPUT.PUT\_LINE('La sede se ha agregado correctamente.');

ELSIF Accion = 'Modificar' THEN

-- Verificar si la sede existe

SELECT COUNT(\*)

INTO Idsede

FROM Sede

Where id\_sede = IdSede;

If IdSede = 0 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'La sede no existe.');

END IF;

-- Modificar la sede

UPDATE Sede

SET nombre\_sede = NombreSede, pais\_Sede = PaisSede, ciudad\_Sede = CiudadSede, avenida\_calle = AvenidaCalle, numero\_sede = NumeroSede, capacidad\_sede = Capacidad

WHERE id\_Sede = IdSede;

DBMS\_OUTPUT.PUT\_LINE('La sede se ha modificado correctamente.');

ELSIF Accion = 'Eliminar' THEN

-- Verificar si la sede existe

SELECT COUNT(\*)

INTO Idsede

FROM Sede

Where id\_sede = IdSede;

If IdSede = 0 THEN

RAISE\_APPLICATION\_ERROR(-20003, 'La sede no existe.');

END IF;

-- Verificar si hay libros asociados a la sede

SELECT COUNT(\*)

INTO Idsede

FROM Material\_bibliografico

Where id\_sede = IdSede;

IF IdSede > 0 THEN

RAISE\_APPLICATION\_ERROR(-20004, 'No se puede eliminar la sede porque existen libros asociados');

END IF;

-- Eliminar la sede

DELETE FROM Sede

WHERE id\_Sede = IdSede;

DBMS\_OUTPUT.PUT\_LINE('La sede se ha eliminado correctamente.');

ELSE

RAISE\_APPLICATION\_ERROR(-20005, 'Acción no válida.');

END IF;

END;

**---PA5. SP\_RegistrarSerial: Este procedimiento se utiliza para registrar nuevos seriales de diarios y revistas en la base de datos.**

CREATE OR REPLACE PROCEDURE SP\_RegistrarSerial (

TipoMaterial IN VARCHAR2,

TituloMaterial IN VARCHAR2,

FechaPublicacion IN DATE,

CodigoSerial IN VARCHAR2,

IdUsuario IN VARCHAR2,

IdEditorial IN VARCHAR2,

IdAdquisicion IN VARCHAR2,

IdProveedor IN VARCHAR2,

IdSede IN VARCHAR2,

IdEstanteria IN VARCHAR2,

IdSeccion IN VARCHAR2,

NumeroPaginas IN NUMBER

) AS

IdMaterialBiblio VARCHAR2(10);

BEGIN

-- Verificar si el serial ya está registrado

SELECT COUNT(\*)

INTO IdMaterialBiblio

FROM Ejemplar

WHERE cod\_serial = CodigoSerial;

IF IdMaterialBiblio > 0 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'El serial ya está registrado.');

END IF;

-- Obtener el ID del material bibliográfico

SELECT id\_material\_biblio

INTO IdMaterialBiblio

FROM Material\_bibliografico

WHERE titulo\_material\_bibliografico = TituloMaterial

FETCH FIRST 1 ROWS ONLY;

IF IdMaterialBiblio IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20002, 'El material bibliográfico no existe.');

END IF;

-- Insertar el serial en la tabla Ejemplar

INSERT INTO Ejemplar (id\_usuario, cod\_serial, id\_material\_biblio, id\_editorial, id\_adquisicion, id\_proveedor, id\_sede, id\_estanteria, id\_seccion, numero\_paginas)

VALUES (IdUsuario, CodigoSerial, IdMaterialBiblio, IdEditorial, IdAdquisicion, IdProveedor, IdSede, IdEstanteria, IdSeccion, NumeroPaginas);

DBMS\_OUTPUT.PUT\_LINE('El serial se ha registrado correctamente.');

END;

**---PA6. SP\_PrestamosActivos: Este procedimiento se utiliza para ver si existen préstamos activos de un determinado libro.**

CREATE OR REPLACE PROCEDURE SP\_PrestamosActivos (

IdMaterialBiblio IN VARCHAR2 -- ID del material bibliográfico a verificar

)

AS

V\_Count NUMBER;

BEGIN

-- Verificar si existen préstamos activos del libro

SELECT COUNT(\*)

INTO V\_Count

FROM Prestamo

WHERE id\_mat\_biblio = IdMaterialBiblio

AND estado\_prestamo = 'prestado';

IF V\_Count > 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Existen préstamos activos del libro.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('No existen préstamos activos del libro.');

END IF;

END;

**PA7. SP\_PrestamoVencido: Este precdimiento se usa regularmente para verificar si hay préstamos vencidos de la tabla PRESTAMO e insertarlos en la tabla MULTAS.**

CREATE OR REPLACE PROCEDURE VerificarPrestamosVencidos

AS

BEGIN

-- Verificar y generar multas para los préstamos vencidos

INSERT INTO Multa (id\_prestamo, id\_usuario, fecha\_imposicion, estado)

SELECT P.id\_prestamo, P.id\_usuario, SYSDATE, 'sin pagar'

FROM Prestamo P

WHERE P.fecha\_devolucion < SYSDATE

AND NOT EXISTS (

SELECT 1

FROM Multa M

WHERE M.id\_prestamo = P.id\_prestamo

);

END;

**Triggers**

**---T1. Trigger\_EjemplarPrestamo: Este TRIGGER se dispara antes de insertar un nuevo pr�stamo verifica que un libro no puede tener una cantidad de ejemplares en pr�stamo mayor que la cantidad total de ejemplares disponibles.**

CREATE OR REPLACE TRIGGER Trigger\_EjemplarPrestamo

AFTER INSERT ON Prestamo

FOR EACH ROW

DECLARE

IdMatBiblio VARCHAR2(10);

AvailableCount NUMBER;

BorrowedCount NUMBER;

BEGIN

IdMatBiblio := :NEW.id\_mat\_biblio;

SELECT COUNT(\*) INTO BorrowedCount

FROM Prestamo P

WHERE P.id\_mat\_biblio = IdMatBiblio

AND P.fecha\_devolucion >= SYSDATE AND estado\_prestamo = 'prestado';

SELECT COUNT(\*) INTO AvailableCount

FROM Ejemplar

WHERE id\_mat\_biblio = IdMatBiblio

AND cod\_serial NOT IN (

SELECT cod\_serial

FROM Prestamo

WHERE id\_mat\_biblio = IdMatBiblio

AND fecha\_devolucion >= SYSDATE AND estado\_prestamo = 'prestado'

);

IF BorrowedCount >= AvailableCount THEN

RAISE\_APPLICATION\_ERROR(-20001, 'No se puede realizar el préstamo. Todos los ejemplares del libro están en préstamo.');

ELSE

NULL;

END IF;

END;

**---T2. Trigger\_PrestamoValidacion: Este trigger se dispara antes de insertar un nuevo pr�stamo en la tabla de pr�stamos. Verifica si el usuario tiene una membres�a activa en la biblioteca antes de permitir el pr�stamo.**

CREATE OR REPLACE TRIGGER Trigger\_PrestamoValidacion

BEFORE INSERT ON Prestamo

FOR EACH ROW

DECLARE

MembresiaCount NUMBER;

BEGIN

-- Verificar si el usuario tiene una membresía activa

SELECT COUNT(\*) INTO MembresiaCount

FROM Usuario U

INNER JOIN Lector L ON U.id\_usuario = L.id\_usuario

WHERE U.id\_usuario = :NEW.id\_usuario

AND L.fecha\_vencimiento\_membresia >= SYSDATE;

IF MembresiaCount > 0 THEN

-- Si el usuario tiene una membresía activa, permitir la inserción del nuevo préstamo

:NEW.estado\_prestamo := 'prestado';

ELSE

-- Si el usuario no tiene una membresía activa, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20001, 'El usuario no tiene una membresía activa. No se puede realizar el préstamo.');

END IF;

END;

**---T3. Trigger\_EjemplarBiblioteca: Este TRIGGER se dispara antes de realizar un pr�stamo. Verifica que un material bibliogr�fico s�lo puede ser prestado si tiene al menos un ejemplar disponible en la biblioteca.**

CREATE OR REPLACE TRIGGER Trigger\_EjemplarBiblioteca

BEFORE INSERT ON Prestamo

FOR EACH ROW

DECLARE

EjemplarCount NUMBER;

BEGIN

-- Verificar si el material bibliográfico tiene al menos un ejemplar disponible en la biblioteca

SELECT COUNT(\*) INTO EjemplarCount

FROM Ejemplar E

INNER JOIN Material\_bibliografico MB ON E.id\_mat\_biblio = MB.id\_mat\_biblio

WHERE MB.id\_mat\_biblio = :NEW.id\_mat\_biblio

AND E.cod\_serial NOT IN (

SELECT cod\_serial

FROM Prestamo

WHERE fecha\_devolucion >= SYSDATE AND estado\_prestamo = 'prestado'

);

IF EjemplarCount > 0 THEN

-- Si hay al menos un ejemplar disponible, permitir la inserción del nuevo préstamo

:NEW.estado\_prestamo := 'prestado';

ELSE

-- Si no hay ejemplares disponibles, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20001, 'No se puede realizar el préstamo. No hay ejemplares disponibles en la biblioteca.');

END IF;

END;

**---T4. Trigger\_MaximosPrestamos: Este TRIGGER se dispara antes de realizar un pr�stamo para verificar que un determinado usuario de la tabla LECTOR no exceda el n�mero m�ximo de libros prestados.**

CREATE OR REPLACE TRIGGER Trigger\_MaximosPrestamos

BEFORE INSERT ON Prestamo

FOR EACH ROW

DECLARE

IdUsuario NUMBER;

Max\_libros\_prestados NUMBER := 5;

PrestamosCount NUMBER;

BEGIN

-- Almacenar el valor del id\_usuario del nuevo registro en una variable

IdUsuario := :NEW.id\_usuario;

-- Verificar el número máximo de libros prestados del usuario

SELECT COUNT(\*) INTO PrestamosCount

FROM Prestamo

WHERE id\_usuario = IdUsuario

AND fecha\_devolucion >= SYSDATE AND estado\_prestamo = 'prestado';

IF PrestamosCount >= Max\_libros\_prestados THEN

-- Si el usuario ha excedido el número máximo de libros prestados, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20001, 'El usuario ha excedido el número máximo de libros prestados.');

ELSE

-- Si el usuario no ha excedido el número máximo de libros prestados, permitir la inserción del nuevo préstamo

:NEW.estado\_prestamo := 'prestado';

END IF;

END;

**---T5. Trigger\_MultaPendiente: Este TRIGGER se dispara antes de realizar un pr�stamo para verificar en la tabla MULTA que el usuario no tiene multas pendientes.**

CREATE OR REPLACE TRIGGER Trigger\_MultaPendiente

BEFORE INSERT ON Prestamo

FOR EACH ROW

DECLARE

IdUsuario NUMBER;

MultasCount NUMBER;

BEGIN

-- Almacenar el valor del id\_usuario del nuevo registro en una variable

IdUsuario := :NEW.id\_usuario;

-- Verificar si el usuario tiene multas pendientes

SELECT COUNT(\*) INTO MultasCount

FROM Multa

WHERE id\_usuario = IdUsuario

AND fecha\_pago IS NULL AND estado = 'sin pagar';

IF MultasCount > 0 THEN

-- Si el usuario tiene multas pendientes, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20001, 'El usuario tiene multas pendientes. No se puede realizar el préstamo.');

ELSE

-- Si el usuario no tiene multas pendientes, permitir la inserción del nuevo préstamo

:NEW.estado\_prestamo := 'prestado';

END IF;

END;

**---T6. Trigger\_FechaDevolucion: Este trigger se activa antes de insertar un nuevo pr�stamo en la tabla de pr�stamos. Verifica que la fecha de devoluci�n sea posterior a la fecha de pr�stamo y realiza ajustes si es necesario.**

CREATE OR REPLACE TRIGGER Trigger\_FechaDevolucion

BEFORE INSERT ON Prestamo

FOR EACH ROW

DECLARE

-- Variables locales para almacenar los valores de inserted

v\_id\_mat\_biblio Prestamo.id\_mat\_biblio%TYPE;

v\_id\_usuario Prestamo.id\_usuario%TYPE;

v\_fecha\_prestamo Prestamo.fecha\_prestamo%TYPE;

v\_fecha\_devolucion Prestamo.fecha\_devolucion%TYPE;

v\_estado\_prestamo Prestamo.estado\_prestamo%TYPE;

BEGIN

-- Almacenar los valores de inserted en las variables locales

v\_id\_mat\_biblio := :NEW.id\_mat\_biblio;

v\_id\_usuario := :NEW.id\_usuario;

v\_fecha\_prestamo := :NEW.fecha\_prestamo;

v\_fecha\_devolucion := :NEW.fecha\_devolucion;

v\_estado\_prestamo := :NEW.estado\_prestamo;

-- Verificar y realizar ajustes en la fecha de devolución si es necesario

IF v\_fecha\_devolucion <= v\_fecha\_prestamo THEN

v\_fecha\_devolucion := v\_fecha\_prestamo + INTERVAL '5' DAY; -- Se agregan 5 días a la fecha de préstamo

END IF;

**END**

**Todos Los triggers**

-- Insertar el nuevo préstamo en la tabla Prestamo

INSERT INTO Prestamo (id\_mat\_biblio, id\_usuario, fecha\_prestamo, fecha\_devolucion, estado\_prestamo)

VALUES (v\_id\_mat\_biblio, v\_id\_usuario, v\_fecha\_prestamo, v\_fecha\_devolucion, v\_estado\_prestamo);

END;

--COMBINACION de los 6 anteriores

CREATE OR REPLACE TRIGGER Trigger\_Prestamo

AFTER INSERT ON Prestamo

FOR EACH ROW

DECLARE

IdUsuario INT;

IdMaterialBiblio VARCHAR(10);

MaxLibrosPrestados INT := 5;

FechaPrestamo DATE;

FechaDevolucion DATE;

PrestamosPendientes INT; -- Variable para almacenar el resultado de la consulta

BEGIN

IdUsuario := :new.id\_usuario;

IdMaterialBiblio := :new.id\_mat\_biblio;

FechaPrestamo := :new.fecha\_prestamo;

-- T4 Verificar el número máximo de libros prestados del usuario

SELECT COUNT(\*)

INTO PrestamosPendientes

FROM Prestamo

WHERE id\_usuario = IdUsuario

AND fecha\_devolucion >= SYSDATE

AND estado\_prestamo = 'prestado';

IF PrestamosPendientes >= MaxLibrosPrestados THEN

-- Si el usuario ha excedido el número máximo de libros prestados, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20001, 'El usuario ha excedido el número máximo de libros prestados.');

END IF;

-- T2 Verificar si el usuario tiene una membresía activa

SELECT COUNT(\*)

INTO PrestamosPendientes

FROM Usuario U

INNER JOIN Lector L ON U.id\_usuario = L.id\_usuario

WHERE U.id\_usuario = IdUsuario

AND L.fecha\_vencimiento\_membresia >= SYSDATE;

IF PrestamosPendientes = 0 THEN

-- Si el usuario no tiene una membresía activa, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20002, 'El usuario no tiene una membresía activa. No se puede realizar el préstamo.');

END IF;

-- T3 y T1 Verificar si el material bibliográfico tiene al menos un ejemplar disponible en la biblioteca

SELECT COUNT(\*)

INTO PrestamosPendientes

FROM Ejemplar E

INNER JOIN Material\_bibliografico MB ON E.id\_mat\_biblio = MB.id\_mat\_biblio

WHERE MB.id\_mat\_biblio = IdMaterialBiblio

AND E.cod\_serial NOT IN (

SELECT cod\_serial

FROM Prestamo

WHERE fecha\_devolucion >= SYSDATE

AND estado\_prestamo = 'prestado'

);

IF PrestamosPendientes = 0 THEN

-- Si no hay ejemplares disponibles, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20003, 'No se puede realizar el préstamo. No hay ejemplares disponibles en la biblioteca.');

END IF;

-- T5 Verificar si el usuario tiene multas pendientes

SELECT COUNT(\*)

INTO PrestamosPendientes

FROM Multa

WHERE id\_usuario = IdUsuario

AND fecha\_pago IS NULL

AND estado = 'sin pagar';

IF PrestamosPendientes > 0 THEN

-- Si el usuario tiene multas pendientes, generar un error y realizar un rollback de la transacción

RAISE\_APPLICATION\_ERROR(-20004, 'El usuario tiene multas pendientes. No se puede realizar el préstamo.');

END IF;

-- T6 Verificar y realizar ajustes en la fecha de devolución si es necesario

FechaDevolucion := CASE

WHEN :new.fecha\_devolucion <= FechaPrestamo THEN FechaPrestamo + 5 -- Se agregan 5 días a la fecha de préstamo

ELSE :new.fecha\_devolucion

END;

-- Actualizar la fecha de devolución en la tabla Prestamo

UPDATE Prestamo

SET fecha\_devolucion = FechaDevolucion

WHERE id\_usuario = IdUsuario

AND id\_mat\_biblio = IdMaterialBiblio;

COMMIT;

END;