Facultad de Ciencias y Tecnología

Ingeniería en Sistemas

Desarrollo de Aplicaciones 6

Proyecto Final

SISTEMA DE ADMINISTRACIÓN PARA BIBLIOTECA EN JAVA CON BASE DE DATOS ORACLE

Docente: Ing. Rolando Gonzales

Estudiantes:

Guzmán Betancur Dennis Adrián

Mendoza Coronado Pedro Michael

Fecha: 03 de agosto del 2020

Santa Cruz - Bolivia

Índice

Documentación

Creación de PDB

```
CREATE PLUGGABLE DATABASE BIBLIOTECABD ADMIN USER adminbiblioteca IDENTIFIED BY b
iblioteca2020 ROLES =(DBA) file_name_convert =(
    'C:\APP\DENNI\PRODUCT\18.0.0\ORADATA\XE\PDBSEED',
    'C:\APP\DENNI\PRODUCT\18.0.0\ORADATA\XE\PDBBIBLIOTECA'
);
ALTER PLUGGABLE DATABASE BIBLIOTECABD OPEN READ WRITE;
ALTER SESSION
SET CONTAINER = BIBLIOTECABD;
```

Creación de Tablespaces

```
CREATE TABLESPACE tsbiblioteca DATAFILE 'C:\APP\DENNI\PRODUCT\18.0.0\ORADATA\XE\P
DBBIBLIOTECA\tsbiblioteca.dbf' size 500k reuse autoextend on next 500k maxsize 5M
;
CREATE TEMPORARY TABLESPACE tstmpbiblioteca tempfile 'C:\APP\DENNI\PRODUCT\18.0.0
\ORADATA\XE\PDBBIBLIOTECA\tstmpbiblioteca.dbf' size 2M reuse autoextend on next 2
M maxsize unlimited;
```

Creación de Roles

```
create role rolprogramador;
grant create session to rolprogramador;
grant alter session to rolprogramador;
grant create trigger to rolprogramador;
grant create view to rolprogramador;
grant create, alter, drop procedure to rolprogramador;
grant select, insert, update, delete on cliente to rolprogramador;
grant select, insert, update, delete on empleado to rolprogramador;
grant select, insert, update, delete on prestamo to rolprogramador;
grant select, insert, update, delete on libro to rolprogramador;
grant select, insert, update, delete on stock to rolprogramador;
grant select, insert, update, delete on autor to rolprogramador;
grant select, insert, update, delete on genero to rolprogramador;
grant select, insert, update, delete on pais_autor to rolprogramador;
grant create profile to rolprogramador;
grant create, alter user to rolprogramador;


create role rolusuario;
grant select, insert, update on libro to rolusuario;
```

```
grant select, insert, update on prestamo to rolusuario;
grant select, insert, update, delete on cliente to rolprogramador;

create role roladministrador;
grant select, insert, update on cliente to rolprogramador;
grant select, insert, update on prestamo to rolprogramador;
grant select, insert, update on libro to rolprogramador;
grant select, insert, update on stock to rolprogramador;
grant select, insert, update on autor to rolprogramador;
grant select, insert, update on genero to rolprogramador;
grant select, insert, update on pais_autor to rolprogramador;
```

Creación de Usuarios

```
create user dbabiblioteca identified by biblioteca2020 default tablespace tsbibli
oteca temporary tablespace tstmpbiblioteca quota unlimited on tsbiblioteca;
grant dba to dbabiblioteca;
-- conn dbabiblioteca/biblioteca2020@localhost:1521/bibliotecabd
alter session
set nls_date_format = 'DD-MON-YYYY HH24:MI:SS';

create user progdennisadrian identified by user2020 default tablespace tsbibliote
ca temporary tablespace tstmpbiblioteca quota unlimited on tsbiblioteca;
grant rolprogramador to progdennisadrian;

create user progpedromendoza identified by user2020 default tablespace tsbibliote
ca temporary tablespace tstmpbiblioteca quota unlimited on tsbiblioteca;
grant rolprogramador to progpedromendoza;

create user admin1 identified by user2020 default tablespace tsbiblioteca tempora
ry tablespace tstmpbiblioteca quota unlimited on tsbiblioteca;
grant roladministrador to admin1;

create user user1 identified by user2020 default tablespace tsbiblioteca temporar
y tablespace tstmpbiblioteca quota unlimited on tsbiblioteca;
grant rolusuario to user1;

create user user2 identified by user2020 default tablespace tsbiblioteca temporar
y tablespace tstmpbiblioteca quota unlimited on tsbiblioteca;
grant rolusuario to user2;
```

Creación de Perfiles

```
CREATE PROFILE perfprogramador limit
FAILED_LOGIN_ATTEMPTS 3
SESSIONS_PER_USER 3
CONNECT_TIME 600
PASSWORD_LOCK_TIME (1/24)/60
PASSWORD_LIFE_TIME 30
PASSWORD_GRACE_TIME 3;

ALTER USER progdennisadrian PROFILE perfprogramador;
ALTER USER progpedromendoza PROFILE perfprogramador;

CREATE PROFILE perfusuario limit
FAILED_LOGIN_ATTEMPTS 5
SESSIONS_PER_USER 1
CONNECT_TIME 480
PASSWORD_LOCK_TIME (1/24)/60
PASSWORD_LIFE_TIME 180
PASSWORD_GRACE_TIME 5;

ALTER USER admin1 PROFILE perfusuario;
ALTER USER user1 PROFILE perfusuario;
ALTER USER user2 PROFILE perfusuario;
```

Creación de Tablas

```
CREATE TABLE pais_autor (
    id int PRIMARY KEY,
    pais varchar2(100) UNIQUE,
    creado_en date,
    modificado_en date
);
CREATE TABLE autor (
    id int PRIMARY KEY,
    nombre varchar2(50),
    id_pais int,
    creado_en date,
    modificado_en date,
    FOREIGN KEY (id_pais) REFERENCES pais_autor(id)
);
CREATE TABLE cliente (
    id int PRIMARY KEY,
    nombre varchar2(20),
```

```
    apellido varchar2(20),
    fecha_nac date,
    domicilio varchar2(100),
    telefono varchar2(20),
    email varchar2(50),
    creado_en date,
    modificado_en date
);
CREATE TABLE genero (
    id int PRIMARY KEY,
    nombre varchar2(30),
    creado_en date,
    modificado_en date
);
CREATE TABLE libro (
    id int PRIMARY KEY,
    nombre varchar2(50),
    isbn varchar2(20),
    anio_publicacion int,
    id_autor int,
    id_genero int,
    creado_en date,
    modificado_en date,
    FOREIGN KEY (id_autor) REFERENCES autor(id),
    FOREIGN KEY (id_genero) REFERENCES genero(id)
);
CREATE TABLE stock (
    id int PRIMARY KEY,
    cantidad int,
    id_libro int,
    creado_en date,
    modificado_en date,
    FOREIGN KEY(id_libro) REFERENCES libro(id)
);
CREATE TABLE empleado (
    id int PRIMARY KEY,
    nombre varchar2(20),
    apellido varchar2(30),
    cargo varchar2(30),
    usuario varchar2(30),
    password varchar2(100),
    fecha_nac date,
    telefono varchar2(50),
    email varchar2(50),
    creado_en date,
```

```
    modificado_en date
);
CREATE TABLE prestamo (
    id int PRIMARY KEY,
    codigo varchar2(10) UNIQUE,
    fecha date,
    devolucion date,
    devuelto char,
    id_libro int,
    id_cliente int,
    id_empleado int,
    creado_en date,
    modificado_en date,
    FOREIGN KEY (id_libro) REFERENCES libro(id),
    FOREIGN KEY (id_cliente) REFERENCES cliente(id),
    FOREIGN KEY (id_empleado) REFERENCES empleado(id)
);
```

Consultas SQL

1. Obtener el ultimo valor actualizado en la tabla “préstamo”

```
select id INTO id_actual from prestamo where devuelto = 'Y' order by devolucion D
ESC fetch first 1 rows only;
```

2. Ver todos los prestamos agrupados por nombre de libro

```
SELECT * FROM prestamo GROUP BY id_libro;
```

3. Ver la información necesaria para libros

```
SELECT l.id,
    l.nombre as nombre_libro,
    l.isbn,
    l.anio_publicacion as publicacion,
    l.id_autor,
    a.nombre as nombre_autor,
    l.id_genero,
    g.nombre as genero,
    s.cantidad
FROM (
        libro l
        JOIN stock s ON (l.id = s.id_libro)
        JOIN autor a ON (l.id_autor = a.id)
```

```
        JOIN genero g ON (l.id_genero = g.id)
    )
```

Vistas

```
CREATE OR REPLACE VIEW vw_empleado_nombre_apellido AS
SELECT
    id,
    UPPER(apellido ||' '|| nombre) as nombre_completo
FROM empleado
ORDER BY apellido ASC;
CREATE OR REPLACE VIEW vw_prestamo_cliente_empleado AS
SELECT p.id,
    p.codigo,
    UPPER(c.apellido ||' '|| c.nombre) as nombre_cliente,
    p.fecha,
    UPPER(l.nombre) as nombre_libro,
    p.devolucion,
    p.devuelto,
    UPPER(e.apellido ||' '|| e.nombre) as nombre_empleado
FROM (
        prestamo p
        JOIN libro l ON (p.id_libro = l.id)
        JOIN cliente c ON (p.id_cliente = c.id)
        JOIN empleado e ON (p.id_empleado = e.id)
    )
ORDER BY p.fecha ASC;

CREATE OR REPLACE VIEW vw_stock_libros AS
SELECT s.id,
    s.cantidad,
    l.nombre
FROM (
        stock s
        JOIN libro l ON (s.id_libro = l.id)
    )
ORDER BY l.nombre ASC;
CREATE OR REPLACE VIEW vw_libros_full AS
SELECT l.id,
    l.nombre as nombre_libro,
    l.isbn,
    l.anio_publicacion as publicacion,
    l.id_autor,
    a.nombre as nombre_autor,
```

```
    l.id_genero,
    g.nombre as genero,
    s.cantidad
FROM (
        libro l
        JOIN stock s ON (l.id = s.id_libro)
        JOIN autor a ON (l.id_autor = a.id)
        JOIN genero g ON (l.id_genero = g.id)
    )
ORDER BY l.nombre ASC;

CREATE OR REPLACE VIEW vw_clientes AS
SELECT s.id,
    s.cantidad,
    l.nombre
FROM (
        stock s
        JOIN libro l ON (s.id_libro = l.id)
    )
ORDER BY l.nombre ASC;
```

Procedimientos Almacenados

```
CREATE OR REPLACE PROCEDURE sp_cliente_insert (
        nombre cliente.nombre%TYPE,
        apellido cliente.apellido%TYPE,
        nacimiento cliente.fecha_nac%TYPE,
        domicilio cliente.domicilio%TYPE,
        telefono cliente.telefono%TYPE,
        email cliente.email%TYPE
    ) AS BEGIN
INSERT INTO cliente (
        id,
        nombre,
        apellido,
        fecha_nac,
        domicilio,
        telefono,
        email,
        creado_en
    )
VALUES (
        0,
        UPPER(nombre),
        UPPER(apellido),
```

```
        nacimiento,
        domicilio,
        telefono,
        email,
        SYSDATE
    );
COMMIT;
END;
/
CREATE OR REPLACE PROCEDURE sp_prestamo_insert(
        codigo prestamo.codigo%TYPE,
        id_libro prestamo.id_libro%TYPE,
        id_cliente prestamo.id_cliente%TYPE,
        id_empleado prestamo.id_empleado%TYPE
    ) AS BEGIN
INSERT INTO prestamo (
        id,
        codigo,
        fecha,
        devuelto,
        id_libro,
        id_cliente,
        id_empleado
    )
VALUES(
        0,
        codigo,
        SYSDATE,
        'N',
        id_libro,
        id_cliente,
        id_empleado
    );
END;
/
CREATE OR REPLACE PROCEDURE sp_devolver_prestamo(
    id_prestamo prestamo.id%TYPE
)
AS
BEGIN
    UPDATE prestamo SET devuelto = 'Y', devolucion = SYSDATE, modificado_en = SYS
DATE WHERE id = id_prestamo;
    COMMIT;
END;
/
```

```
CREATE OR REPLACE PROCEDURE sp_prestamo_por_apellido_fecha_select(
    nombre IN cliente.nombre%TYPE,
    fecha_inicial prestamo.fecha%TYPE,
    fecha_final prestamo.devolucion%TYPE
)
AS
    c1 SYS_REFCURSOR;
BEGIN
    open c1 for
    select * from vw_prestamo_cliente_empleado where nombre_cliente LIKE UPPER('%
' || nombre || '%') AND fecha >= fecha_inicial AND fecha <= fecha_final ORDER BY
nombre_cliente;
    DBMS_SQL.RETURN_RESULT(c1);
END;
/
CREATE OR REPLACE PROCEDURE sp_prestamo_por_libro_fecha_select(
    nombre IN libro.nombre%TYPE,
    fecha_inicial prestamo.fecha%TYPE,
    fecha_final prestamo.fecha%TYPE
)
AS
    c1 SYS_REFCURSOR;
BEGIN
    open c1 for
    select * from vw_prestamo_cliente_empleado where nombre_libro LIKE UPPER('%'
|| nombre || '%') AND fecha >= fecha_inicial AND fecha <= fecha_final ORDER BY no
mbre_cliente;
    DBMS_SQL.RETURN_RESULT(c1);
END;
/
CREATE OR REPLACE PROCEDURE sp_cliente_nombre_apellido_select(
    nombre_cliente IN cliente.nombre%TYPE
)
AS
    c1 SYS_REFCURSOR;
BEGIN
    open c1 for
    select * from vw_clientes_nombre_apellido where nombre_completo LIKE UPPER('%
' || nombre_cliente || '%') AND rownum <= 50;
    DBMS_SQL.RETURN_RESULT(c1);
END;
/
```

Triggers

```
CREATE OR REPLACE TRIGGER TRG_PRESTAMO_SEQ BEFORE
INSERT ON prestamo REFERENCING new as new for each row
declare proximoid Number := 0;
begin
select seq_prestamo.nextval into proximoid
from dual;
:new.id := proximoid;
end;
/
CREATE OR REPLACE TRIGGER TRG_DISMINUIR_STOCK
AFTER
INSERT ON prestamo
DECLARE
    id_actual Number := 0;
    id_libro_insertado Number := 0;
    stock_actual Number := 0;
BEGIN
    SELECT seq_prestamo.CURRVAL INTO id_actual FROM dual;
    SELECT id_libro INTO id_libro_insertado FROM prestamo WHERE id = id_actual;
    UPDATE stock SET cantidad = (cantidad - 1) WHERE id_libro = id_libro_insertad
o;
END;
/
CREATE OR REPLACE TRIGGER TRG_AUMENTAR_STOCK
AFTER
UPDATE ON prestamo
DECLARE
    id_actual Number := 0;
    id_libro_devuelto Number := 0;
    stock_actual Number := 0;
BEGIN
    select id INTO id_actual from prestamo where devuelto =  'Y' order by devoluc
ion DESC fetch first 1 rows only;
    SELECT id_libro INTO id_libro_devuelto FROM prestamo WHERE id = id_actual;
    UPDATE stock
    SET cantidad = (cantidad + 1)
    WHERE id_libro = id_libro_devuelto;
END;
/
CREATE OR REPLACE TRIGGER TRG_CREATE_STOCK
AFTER INSERT
    ON libro
    FOR EACH ROW
```

```
BEGIN
    INSERT INTO stock
    ( cantidad,
    id_libro,
    creado_en)
    VALUES
    ( 0,
    :new.id,
    SYSDATE);
END;
/
```

Índices

```
CREATE INDEX idx_cliente_apellido_nombre ON cliente (apellido, nombre);
CREATE INDEX idx_libros_nombre ON libro (nombre);
CREATE INDEX idx_prestamos ON prestamo (fecha, devolucion);
CREATE INDEX idx_empleados ON empleado (apellido, nombre);
```

Secuencias

```
CREATE SEQUENCE seq_pais increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_autor increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_cliente increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_genero increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_libro increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_stock increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_empleado increment by 1 start with 1 nocache;
CREATE SEQUENCE seq_prestamo increment by 1 start with 1 nocache;
```