Integrante: Luis Villarroel

C.I: 16.313.858

Tarea 1.4: Creación de vistas:

Crear vistas para simplificar consultas complejas y proporcionar acceso restringido a ciertos datos.

- Implementar seguridad y escalabilidad acordes a las necesidades de la biblioteca.

Este esquema puede adaptarse en función de los requerimientos específicos o la evolución de los mismos en el futuro.

Realizar esta actividad a la base de datos: Consideraciones Adicionales:

● Seguridad: Implementar medidas de seguridad robustas para proteger la información

confidencial y evitar el acceso no autorizado.

● Escalabilidad: Diseñar un sistema escalable para hacer frente al crecimiento continuo de la

biblioteca y el número de usuarios.

● Colaboración: Fomentar la colaboración con otras bibliotecas cósmicas para compartir

recursos y conocimientos

Para abordar adecuadamente las consideraciones adicionales de seguridad, escalabilidad y colaboración en el diseño de la base de datos de la biblioteca, es necesario implementar varias estrategias y técnicas en el entorno de PostgreSQL. A continuación, se presentan recomendaciones y enfoques específicos para cada consideración, junto con ejemplos de cómo implementarlos en la base de datos.

1. Seguridad

a. Roles y Permisos:

• Utilizar roles para gestionar el acceso a la base de datos y asegurar que solo los usuarios autorizados tengan acceso a funciones específicas.

Ejemplo:

```
-- Crear rol para administradores
CREATE ROLE admin_role;

-- Crear rol para usuarios
CREATE ROLE user_role;

-- Conceder permisos a los roles
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin_role;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public TO user_role;
```

b. Autenticación de Usuarios:

- Usar métodos de autenticación robustos como la autenticación de dos factores (2FA) o acceso basado en LDAP (Lightweight Directory Access Protocol).

c. Encriptación:

- Asegurar que los datos sensibles, como contraseñas de usuarios y correos electrónicos, estén encriptados. Usar la función pgcrypto de PostgreSQL para encriptación.

Ejemplo de encriptación de contraseñas:

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Almacenar contraseñas encriptadas
INSERT INTO usuarios (nombre, apellido, email, password)
VALUES ('Juan', 'Pérez', 'juan.perez@example.com', crypt('MiContraseñaSegura',
gen_salt('bf')));
```

2. Escalabilidad

a. Diseño de la Estructura:

- Asegurarse de que las tablas y los índices estén bien diseñados para soportar un gran volumen de datos. Implementar índices donde sea necesario para acelerar las consultas.

Ejemplo:

```sql
-- Crear un índice para acelerar las búsquedas por título
CREATE INDEX idx_libros_titulo ON libros(titulo);
```

b. Particionamiento:

- Considerar el uso del particionamiento de tablas para dividir tablas grandes en partes más manejables, lo que facilita las consultas y mejora el rendimiento.

Ejemplo:

```sql
CREATE TABLE prestamos_2022 PARTITION OF prestamos FOR VALUES FROM ('2022-01-01') TO ('2023-01-01');
CREATE TABLE prestamos_2023 PARTITION OF prestamos FOR VALUES FROM ('2023-01-01') TO ('2024-01-01');
```

- Configurar la replicación de bases de datos para asegurar la alta disponibilidad y distribuir la carga, permitiendo que las lecturas se realicen desde las réplicas en lugar de la base de datos principal.

3. Colaboración

a. API para Acceso Externo:

- Implementar una API (por ejemplo, REST API) que permita a otras bibliotecas acceder a ciertos recursos, como libros y autorizaciones de préstamo.

b. Compartición de Datos:

- Diseñar una tabla para registrar colaboraciones con otras bibliotecas, permitiendo compartir recursos y conocimientos (por ejemplo, listas de libros en coautoría).

Ejemplo:

```sql
CREATE TABLE colaboraciones (
    id SERIAL PRIMARY KEY,
    id_biblioteca INT,
    libro_id INT REFERENCES libros(id),
    fecha_colaboracion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    detalles TEXT
);
```

c. Cumplimiento de Normativa:

- Asegurar que todas las medidas de colaboración cumplan con las regulaciones locales de privacidad y protección de datos (por ejemplo, GDPR o las regulaciones de protección de datos en su región).

Código Completo para Consideraciones Adicionales

Aquí está un código SQL compacto que implementa estas recomendaciones:

```sql
-- 1. Seguridad
CREATE ROLE admin_role;
CREATE ROLE user_role;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO admin_role;
GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public TO user_role;

-- 2. Encriptación de contraseñas
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- Ejemplo de inserción con contraseña encriptada
INSERT INTO usuarios (nombre, apellido, email, password)
VALUES ('Juan', 'Pérez', 'juan.perez@example.com', crypt('MiContraseñaSegura',
gen_salt('bf')));

-- 3. Escalabilidad
CREATE INDEX idx_libros_titulo ON libros(titulo);

-- Ejemplo de particionamiento de tabla
CREATE TABLE prestamos_2022 PARTITION OF prestamos FOR VALUES FROM ('2022-01-
01') TO ('2023-01-01');
CREATE TABLE prestamos_2023 PARTITION OF prestamos FOR VALUES FROM ('2023-01-
01') TO ('2024-01-01');
```

```sql
-- 4. Colaboración
CREATE TABLE colaboraciones (
    id SERIAL PRIMARY KEY,
    id_biblioteca INT,
    libro_id INT REFERENCES libros(id),
    fecha_colaboracion TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    detalles TEXT
);
```

Con estas implementaciones, la base de datos no solo protegerá la información sensible frente a accesos no autorizados, sino que estará diseñada para escalar conforme crezca la biblioteca y facilitará la colaboración con otras entidades. Estas características son esenciales para ofrecer un servicio eficiente y seguro que cumpla con las expectativas de los usuarios y las normativas vigentes.