Creación del código SQL para la base de datos "biblioteca”

Investigaciones

Objetivo: Desarrollar un sistema digital para la gestión de préstamos de libros, usuarios y reportes, combinando estructuras de datos avanzadas en Python con una base de datos relacional MySQL normalizada hasta 3FN.

1) Se investigaron los conceptos de normalización (1FN, 2FN y 3FN) para garantizar una base de datos libre de redundancias y con integridad referencial.

2) Se evaluaron estructuras de datos en Python para aumentar eficiencia:
Árbol binario de búsqueda (ABB) para localizar libros rápidamente.
Lista enlazada para manejar préstamos activos.

3) Cola (FIFO) para controlar vencimientos por fecha.
Pila (LIFO) para historial de devoluciones.

4) Se revisaron consultas SQL para generar reportes automatizados:
Libros más prestados, libros con baja disponibilidad y préstamos vencidos.

Procesos explicativos

1)

Explicación de 1FN:

1FN primera forma normal deberá contener un valor atómico osea indivisible, que no puede descomponer en partes más pequeñas dentro de una tabla o columna. La columna deberá de tener un único tipo de dato. En una base de datos, cada campo o columna debe contener ***solo un dato*** *por registro, no varios datos juntos.*

Explicación de 2FN:

La tabla debe estar en f1, Todos los atributos deben depender de toda la clave primaria.No debe haber campos con listas de valores. Ejemplo de error: "Producto: mouse, teclado, monitor"

Explicación de 3FN:

La tercera forma normal debe de estar en 2FN para que pueda funcionar y no debe de haber dependencias transitivas que seria de A—> B y B—> C es mejor evitarlo haciendo que de A—> C

2)se usaron las siguientes estructuras de datos para aumentar la eficiencia.

Listas:

Estructura básica y muy flexible.

Permite acceso por índice en O(1).

Operaciones como agregar al final(append) son O(1) en promedio.

Útil para almacenar colecciones ordenadas y modificar elementos.

Diccionarios:

Colección de pares clave-valor.

Búsqueda, inserción y eliminación en O(1) promedio.

Ideal para mapas y accesos rápidos por clave.

*Tuplas**:***

Son listas inmutables.

Usadas cuando necesitas que los datos no cambian, pueden ser más eficientes que listas.

Buenas para claves en diccionarios o para devolver múltiples valores.

3)

Cola (FIFO) Cola para controlar vencimiento por fecha: *ejemplo, es como una fila de personas esperando. El primero que llega es el primero que se atiende. Se usa para controlar vencimientos, porque el primero en entrar es el primero en vencer. Así podés ver qué cosas vencer antes.*

Pila (LIFO) para historial de devoluciones: *ejemplo, es como pila de platos, donde el último plato que pusiste es el primero que sacás. Se usa para llevar el historial de devoluciones, porque lo último que devolves es lo primero que revisas.*

4) <u>Consultas:</u>

Utilizamos las consultas para eliminar, modificar, limitar o ordenar datos que ya están insertados en la base de datos. Para esto necesitamos saber qué son las instrucciones de SQL y para cuándo usarlas:
UPDATE *para modificar registros existentes en una tabla.*

SELECT *se usa para consultar (obtener) datos de una tabla o varias tablas.*

DELETE *lo usamos para eliminar registros de una tabla.*

LIMIT *que se usaría para limitar el número de filas devueltas por la consulta.*

ORDER BY *utilizado para ordenar los resultados de la consulta en función de una o más columnas.*

Para libros más prestados utilizaremos:

```
SELECT titulo, autor FROM libros, prestamos
WHERE id_libro = id_libro
GROUP BY id_libro
ORDER BY veces_prestados DESC;
```

Para libros con baja disponibilidad utilizamos:

```
SELECT titulo, autor, cantidad_total
FROM libro, prestamos
WHERE id_libro = id_libro AND fecha_vencimiento
ORDER BY id_libro
HAVING disponibles <2;
```

Para préstamos vencidos utilizamos:

```
SELECT nombre, titulo, fecha_prestamos, fecha_vencimiento
FROM usuarios, prestamos, libros
WHERE id_usuarios = id_usuarios AND id_libro = id_libro AND fecha_devolucion AND
fecha_vencimiento < CURDATE();
```