Autor: Daniel Simonetta
Comisión: 81805

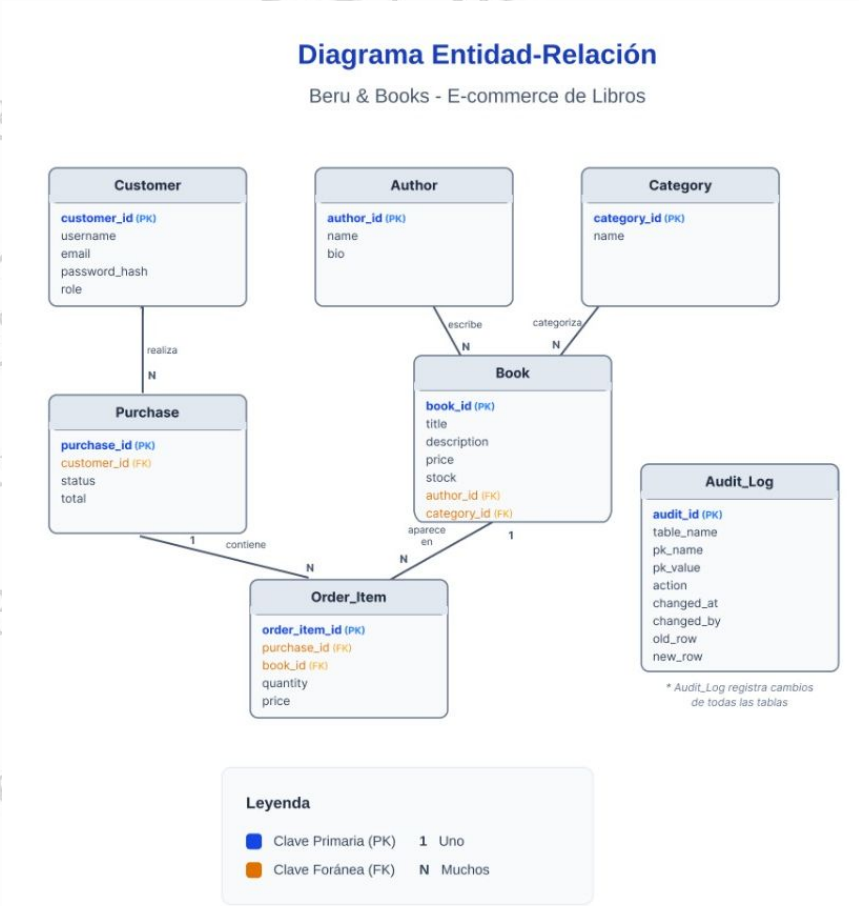
Problematica:

En el contexto actual, muchas librerías digitales carecen de un sistema unificado para manejar usuarios, libros, autores y pedidos. Esto genera errores en la gestión de stock, duplicación de datos y dificultades para obtener reportes de ventas.

El proyecto Beru & Books busca resolver estas brechas mediante una base de datos relacional que centraliza toda la información de clientes, catálogo y transacciones.

Diagrama Entidad Relación:

Representa las entidades principales del sistema (usuarios, autores, categorías, libros, pedidos e items de pedido) y sus relaciones.



Listado de Tablas:

A continuación se detallan las tablas de la base de datos Beru & Books:

Tabla Users

Campo	Tipo Dato	PK	FK	NULL
user_id	INT	X		
username	VARCHAR(50) (UNIQUE)			
email	VARCHAR(100) (UNIQUE)			
password_hash	VARCHAR(255)			
role	ENUM(customer, admin)			
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

Tabla Authors

Campo	Tipo Dato	PK	FK	NULL
author_id	INT	X		
name	VARCHAR(100)			
bio	TEXT			

Tabla Categories

Campo	Tipo Dato	PK	FK	NULL
category_id	INT	X		
name	VARCHAR(100) UNIQUE			

Tabla Books

Campo	Tipo Dato	PK	FK	NULL
book_id	INT	X		
title	VARCHAR(200)			
description	TEXT			
price	DECIMAL(10,2)			
stock	INT			
author_id	INT		X	
category_id	INT		X	
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

Tabla Orders

Campo	Tipo Dato	PK	FK	NULL
order_id	INT	X		
user_id	INT		X	
status	ENUM(pending,paid,shipped,cancelled)			
total	DECIMAL(10,2)			
created_at	TIMESTAMP			
updated_at	TIMESTAMP			

Tabla Order_Items

Campo	Tipo Dato	PK	FK	NULL
order_item_id	INT	X		
order_id	INT		X	
book_id	INT		X	
quantity	INT (CHECK > 0)			
price	DECIMAL(10,2)			

Vistas creadas

En esta segunda etapa del proyecto *Beru & Books*, se incorporaron diferentes vistas dentro de la base de datos. El objetivo principal fue facilitar el acceso a la información más relevante del sistema, evitando consultas complejas y mejorando la legibilidad de los reportes.

1. `vw_books_with_authors`
Esta vista une los datos de las tablas **book**, **author** y **category**.
Permite obtener, en una sola consulta, el título de cada libro, su precio, el nombre del autor y la categoría a la que pertenece.
El objetivo es simplificar las consultas que se realizan en la interfaz de administración o en los reportes de catálogo, sin tener que escribir múltiples `JOIN`.

2. `vw_purchase_details`
Esta vista combina las tablas **purchase**, **customer**, **order_item** y **book**, mostrando el detalle completo de cada compra: qué cliente la realizó, el estado del pedido, los productos incluidos, sus cantidades y precios unitarios.
Su uso principal es generar reportes administrativos o comprobantes internos de ventas, integrando toda la información del proceso de compra en una sola consulta.

3. `vw_book_stock_simple`
Una vista más sencilla que devuelve únicamente el `book_id`, el título y el stock actual.
Se creó con el objetivo de ofrecer una manera rápida de controlar el inventario sin cargar datos adicionales. Es útil para paneles internos o scripts automáticos que verifican disponibilidad de productos.

Funciones creadas

Las funciones agregadas tienen como propósito encapsular operaciones frecuentes dentro de la base de datos y devolver un único valor calculado, mejorando así la consistencia y el rendimiento de las consultas.

1. `fn_book_stock(p_book_id)`
Recibe como parámetro el identificador de un libro (`book_id`) y devuelve el stock actual de ese título.
Se utiliza principalmente en reportes y validaciones internas, por ejemplo, antes de realizar una venta o actualizar una orden.
Manipula la tabla **book**, de la cual obtiene directamente el valor de la columna `stock`.

2. `fn_purchase_total(p_purchase_id)`
Calcula el total de una compra a partir de los ítems registrados en la tabla **order_item**.
Recibe el ID del pedido (`purchase_id`), multiplica la cantidad por el precio de cada producto y devuelve la suma total.
El objetivo de esta función es garantizar que el total de una compra se pueda recalcular automáticamente de forma precisa, sin depender de cálculos externos.

Stored Procedures creados

Los procedimientos almacenados fueron implementados para automatizar operaciones frecuentes, optimizar la gestión de pedidos y reducir errores humanos durante los procesos de carga o actualización de datos.

1. `sp_add_order_item`

Este procedimiento permite agregar un nuevo ítem dentro de una compra existente.

Primero verifica que haya stock disponible del libro seleccionado, luego inserta el registro en la tabla **order_item**, actualiza el stock del libro en la tabla **book**, y recalcula automáticamente el total de la compra en **purchase**.

Su función principal es mantener la integridad de los datos al momento de registrar una venta.

2. `sp_recalculate_purchase_total`

Se usa para recalcular el total de una compra cuando se modifican los ítems asociados (por ejemplo, al eliminar o actualizar un producto).

Actualiza el campo `total` en la tabla **purchase** a partir de los valores actuales de **order_item**.

Este procedimiento garantiza que los montos de las órdenes siempre estén actualizados sin necesidad de operaciones manuales.

Script en SQL de creación de la base de datos y tablas.