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.

MODELO DE NEGOCIO

Beru & Books es una plataforma argentina de venta y distribución de libros, inspirada en modelos de e-commerce como Amazon o Mercado Libre, pero centrada exclusivamente en la literatura nacional.

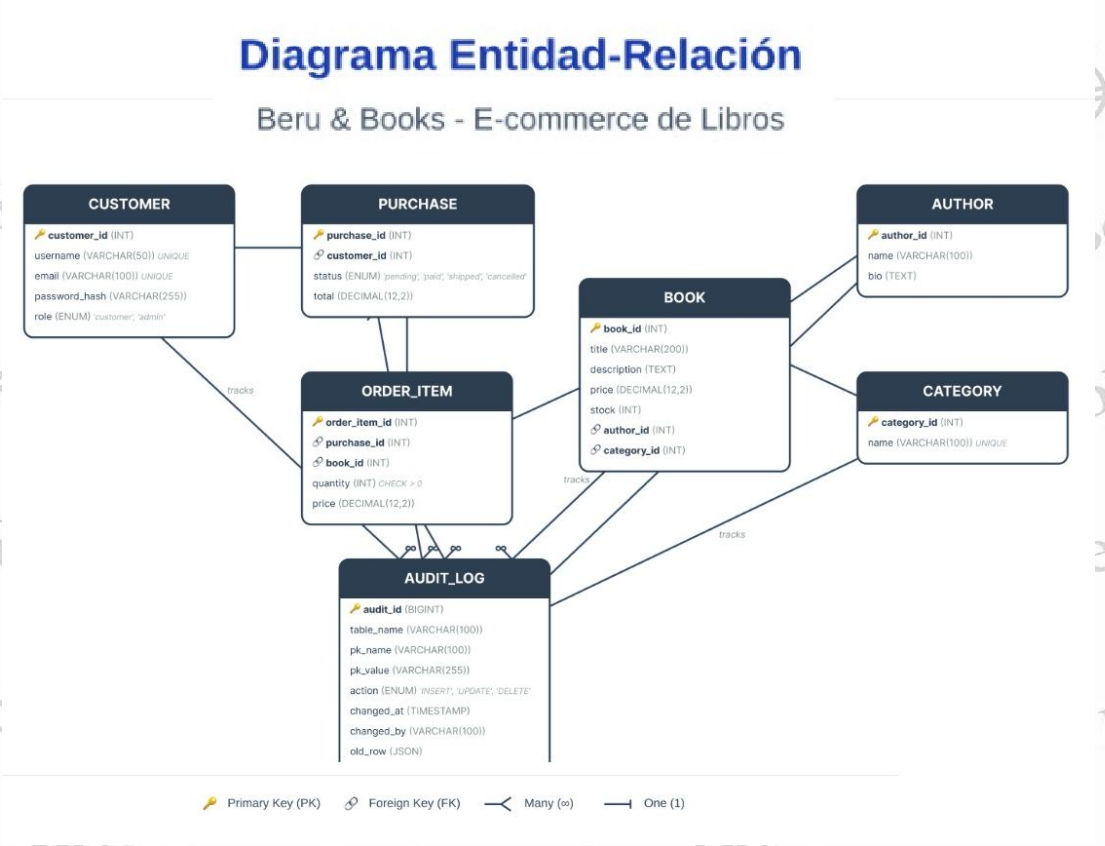
El objetivo del negocio es conectar autores, editoriales independientes y lectores, permitiendo comprar libros en formato físico y digital.

El modelo contempla el registro de usuarios, gestión de catálogo de libros, procesamiento de compras, control de stock y trazabilidad de transacciones a través de un sistema de auditoría.

La base de datos fue diseñada para garantizar integridad referencial, escalabilidad y trazabilidad completa de cada operación, contemplando aspectos administrativos (pedidos, usuarios, autores, categorías), operativos (stock, ventas) y analíticos (informes de ventas y auditorías).

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

Las vistas se diseñaron con el objetivo de simplificar consultas complejas, reunir información de distintas tablas y facilitar la generación de reportes para análisis y control operativo.

1. **vw_books_with_authors**
   Esta vista muestra todos los libros del catálogo junto con su autor y categoría correspondiente.
   Combina datos de las tablas **book**, **author** y **category** mediante `JOIN`, lo que permite obtener una vista unificada del catálogo editorial.
   Su uso principal es facilitar las búsquedas y listados del catálogo de libros disponibles.

2. **vw_purchase_details**
   Presenta información completa sobre cada compra, incluyendo el nombre del cliente, los libros adquiridos, cantidades, precios y estado del pedido.
   Se construye a partir de las tablas **purchase**, **customer**, **order_item** y **book**.
   Su objetivo es agilizar los reportes de ventas y permitir el seguimiento de pedidos individuales desde una única consulta.

3. **vw_customer_orders_summary**
   Muestra un resumen por cliente, con el total de compras realizadas y el monto total gastado.
   Se compone de las tablas **customer** y **purchase**, agrupando resultados por cliente.
   Esta vista sirve como base para análisis de comportamiento de clientes y reportes de fidelización.

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_list_books**
   Este procedimiento devuelve un listado completo del catálogo de libros, incluyendo el autor y la categoría de cada uno.
   Reúne información de las tablas **book**, **author** y **category**, ordenando los resultados alfabéticamente por título.
   Su objetivo es brindar una consulta rápida y estructurada para interfaces o reportes de administración.

2. **sp_customer_purchases**
   Permite obtener todas las compras realizadas por un cliente en particular.
   Combina las tablas **purchase** y **order_item**, mostrando el número de pedido, estado, total y cantidad de ítems.
   Es útil para reportes personalizados y para el historial de compras en el panel del cliente.

3. **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 **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.

4. **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.

Triggers creados

Los *triggers* se implementaron para reforzar la integridad de los datos y asegurar que las operaciones críticas (como altas, bajas o actualizaciones de pedidos) se registren automáticamente en la tabla de auditoría (*audit_log*).
Además, algunos de ellos mantienen la coherencia del stock de libros frente a las ventas o eliminaciones.

1. **trg_order_item_after_insert**
   Este *trigger* se ejecuta inmediatamente después de insertar un nuevo registro en la tabla **order_item**.
   Su función es doble: por un lado, actualiza el stock del libro correspondiente en la tabla **book**, reduciendo la cantidad disponible según el número de unidades vendidas; por otro, inserta un registro en la tabla **audit_log**, guardando el detalle del cambio realizado.
   De esta forma, cada venta queda registrada y el inventario se actualiza automáticamente.

2. **trg_order_item_after_delete**
   Se activa tras eliminar un registro de la tabla **order_item**.
   Su tarea principal es restaurar el stock del libro que había sido reducido en la venta original, garantizando que no haya pérdidas o desbalances en el inventario.
   También registra el evento en la tabla **audit_log**, dejando constancia de la eliminación del ítem en cuestión.

Informes generados

Las vistas y procedimientos fueron pensados para permitir informes y análisis de datos.
Algunos ejemplos:

- **Ventas por autor y categoría:** obtenidas a partir de `vw_books_with_authors` y `vw_purchase_details`.
- **Resumen de clientes y montos gastados:** desde `vw_customer_orders_summary`.
- **Control de stock:** mediante la función `fn_book_stock` y la vista `vw_book_stock`.
- **Auditoría de cambios:** gracias a los triggers que registran toda acción en `audit_log`.

Estos informes permiten analizar el rendimiento de ventas, identificar autores más vendidos y detectar modificaciones en datos sensibles.

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