Online Bookstore Database Project

Project Overview

Objective

The goal of this project is to design a relational database for an online bookstore that efficiently manages data related to customers, books, authors, orders, and order details. The database will enforce data integrity, optimize performance, and support future scalability.

Key Features

- **Data Integrity**: Enforced through relationships, constraints, and cascading actions.
- **Optimized Performance**: Use of indexing and efficient schema design.
- **Scalability**: Structured to handle growing datasets and functionalities.
- **Maintainability**: Clear documentation and modular design for easy updates.

Functional Requirements

1. **Authors Management**:
   - Store details of authors, ensuring unique identification via email.
   - Maintain relationships between authors and their books.
2. **Books Inventory**:
   - Keep track of books with details like price, stock quantity, and associated author.
   - Support updates to stock levels and pricing.
3. **Customers Management**:
   - Store customer information, ensuring unique identification via email.
   - Enable tracking of customer orders and order history.
4. **Orders Management**:
   - Record customer orders with details like order date, total amount, and order line items.
   - Calculate line totals automatically based on book price and quantity.
5. **Order Details**:

- Maintain relationships between orders and books with quantity and computed line totals.
- Support cascading deletions when orders are removed.

Requirement Analysis

Entities and Attributes

Below are the entities required for this project along with their respective attributes:

1. **Authors**
    - `author_id` : Primary Key (INT, Auto-Increment)
    - `name` : Author's name (VARCHAR, NOT NULL)
    - `email` : Author's email (VARCHAR, UNIQUE, NOT NULL)
2. **Books**
    - `book_id` : Primary Key (INT, Auto-Increment)
    - `title` : Book title (VARCHAR, NOT NULL)
    - `price` : Book price (DECIMAL, NOT NULL)
    - `stock_quantity` : Number of books in stock (INT, DEFAULT 0)
    - `author_id` : Foreign Key referencing `Authors.author_id`
3. **Customers**
    - `customer_id` : Primary Key (INT, Auto-Increment)
    - `name` : Customer's name (VARCHAR, NOT NULL)
    - `email` : Customer's email (VARCHAR, UNIQUE, NOT NULL)
    - `phone` : Customer's phone number (VARCHAR, NOT NULL)
4. **Orders**
    - `order_id` : Primary Key (INT, Auto-Increment)
    - `customer_id` : Foreign Key referencing `Customers.customer_id`
    - `order_date` : Date and time of the order (DATETIME, Default: CURRENT_TIMESTAMP)
    - `total_amount` : Total amount of the order (DECIMAL, NOT NULL)
5. **OrderDetails**
    - `order_details_id` : Primary Key (INT, Auto-Increment)
    - `order_id` : Foreign Key referencing `Orders.order_id`
    - `book_id` : Foreign Key referencing `Books.book_id`
    - `quantity` : Number of books ordered (INT, NOT NULL)

- `line_total`: Computed field (DECIMAL) = `quantity * book price`

Relationships

1. **Authors → Books**:
    - One-to-Many relationship.
    - A book must be associated with an author.
2. **Customers → Orders**:
    - One-to-Many relationship.
    - An order must belong to a customer.
3. **Orders → OrderDetails**:
    - One-to-Many relationship.
    - Each order can have multiple order details (line items).
4. **Books ↔ OrderDetails**:
    - Many-to-Many relationship (via `OrderDetails`).
    - A book can appear in multiple orders, and an order can contain multiple books.

Cascading and Restricting Needs

1. Authors → Books

- **On Update**: CASCADE (e.g., if `author_id` changes, update in `Books`).
- **On Delete**: SET NULL (retain the book record but set `author_id` to NULL).

2. Customers → Orders

- **On Update**: CASCADE (reflect changes in customer details in their orders).
- **On Delete**: RESTRICT (prevent customer deletion if they have orders).

3. Orders → OrderDetails

- **On Update**: CASCADE (reflect changes in order details).
- **On Delete**: CASCADE (delete associated order details if an order is deleted).

4. Books → OrderDetails

- **On Update**: CASCADE (reflect book changes in associated order details).

- **On Delete**: RESTRICT (prevent deletion if a book is part of an order).

Next Step: ER Diagram

The next step is to create an ER diagram based on this analysis. The diagram will visually represent the entities, attributes, and relationships defined above. This can be created using tools like Lucidchart, draw.io, or dbdiagram.io