BookWise

Oracle SQL Database Project

CSE 3110: Database Systems Laboratory

Abdullah Al Shafi
Roll : 2007055
Dept. of CSE
KUET

Project Introduction

BookWise stands as a revolutionary book management system tailored to cater to the diverse needs of book enthusiasts, authors, publishers, and sellers. Functioning as a centralized solution, it leverages a robust Oracle SQL database to streamline various aspects of the book industry.

Project Goals

I. Comprehensive Database Schema Design :

The primary aim is to devise a relational database schema capable of efficiently storing and organizing extensive book-related data. This encompasses books, authors, publishers, genres, sales, and customer information, ensuring seamless data retrieval and management across various book-related processes.

II. Implementation of SQL Queries:

The project seeks to implement a diverse range of SQL queries to facilitate data manipulation and retrieval within the database. These queries enable functionalities such as adding new books, updating records, and retrieving detailed information based on specific criteria.

Significance of Database in Book Management

In the realm of book management, a meticulously structured database plays an indispensable role. It serves as the backbone for efficient inventory management, sales tracking, author details, and customer engagement. The database empowers informed decision-making, enhances book discovery for readers, and optimizes operational efficiency.

Database Schema Overview:

The foundation of the BookWise database revolves around several core tables meticulously designed to encompass all essential book-related data:

- **Books:** This table stores comprehensive details about individual books, including title, author, genre, price, publication date, and inventory status.
- **Customers:** It contains pertinent data about book customers, including their name, contact details, and address.
- **Genre:** This table stores genre information to categorize books into different genres.
- **Inventory:** It tracks book inventory, including quantity and availability.
- **Sales:** This table records sales details, including customer information, sale date, total price, and employee involved.
- **Employees:** It contains employee data relevant to book sales and management.
- **Order:** This table tracks customer book orders, including order date and quantity.
- **Authors:** It stores detailed information about book authors, including their name, nationality, and birth/death dates.

Design Rationale:

The database schema for BookWise is meticulously designed to efficiently manage book-related data, catering to various aspects such as book details, sales tracking, and author management. Each table serves a specific purpose, ensuring clarity and organization while facilitating seamless data retrieval and manipulation. Through strategic use of foreign keys, the schema promotes data integrity and relational connections, minimizing the risk of inconsistencies.

Overall, the schema provides a robust foundation for book management operations, supporting the diverse needs of book enthusiasts, authors, publishers, and sellers within the dynamic book industry ecosystem.

Table Relationships:

The relationships between tables are primarily defined by foreign keys to ensure data integrity and relational connections. For instance:

```
-- Books table
CREATE TABLE Books (
    book_id INT PRIMARY KEY,
    title VARCHAR(255),
    author VARCHAR(255),
    genre_id INT,
    price DECIMAL(10, 2),
    publication_date DATE,
    FOREIGN KEY (genre_id) REFERENCES Genre(genre_id) ON DELETE CASCADE
);
```

```
-- Customers table
CREATE TABLE Customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    address VARCHAR(255)
);
```

```
-- Genre table
CREATE TABLE Genre (
    genre_id INT PRIMARY KEY,
    name VARCHAR(100)
);
```

```sql
-- Inventory table
CREATE TABLE Inventory (
    inventory_id INT PRIMARY KEY,
    book_id INT,
    quantity INT,
    available BOOLEAN,
    FOREIGN KEY (book_id) REFERENCES Books(book_id) ON DELETE CASCADE
);
```

```sql
-- Sales table
CREATE TABLE Sales (
    sale_id INT PRIMARY KEY,
    customer_id INT,
    book_id INT,
    sale_date DATE,
    total_price DECIMAL(10, 2),
    employee_id INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE
CASCADE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id) ON DELETE CASCADE,
    FOREIGN KEY (employee_id) REFERENCES Employees(employee_id) ON DELETE
CASCADE
);
```

```sql
-- Employees table
CREATE TABLE Employees (
    employee_id INT PRIMARY KEY,
    name VARCHAR(255),
    email VARCHAR(255),
    phone_number VARCHAR(20),
    position VARCHAR(100),
    hire_date DATE
);
```

```
-- Order table
CREATE TABLE Order (
    order_id INT PRIMARY KEY,
    customer_id INT,
    book_id INT,
    order_date DATE,
    quantity INT,
    FOREIGN KEY (customer_id) REFERENCES Customers(customer_id) ON DELETE
CASCADE,
    FOREIGN KEY (book_id) REFERENCES Books(book_id) ON DELETE CASCADE
);
```

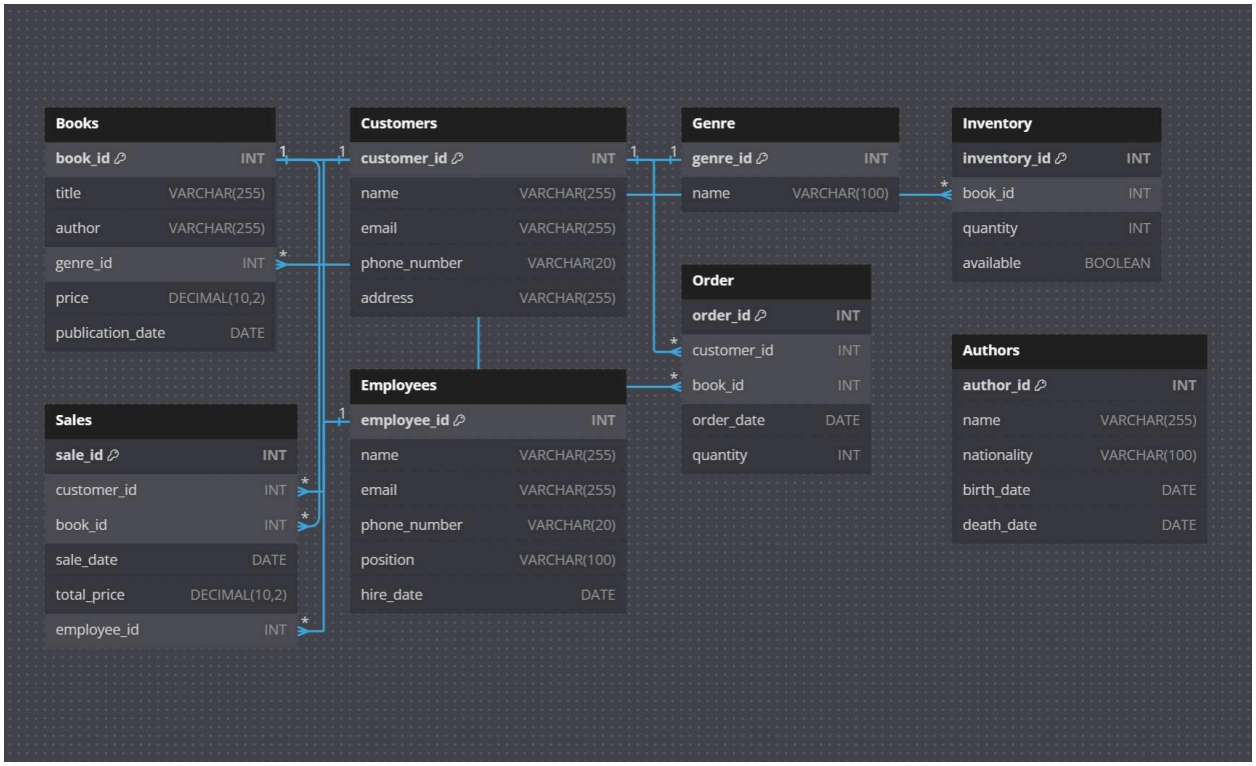
```
-- Authors table
CREATE TABLE Authors (
    author_id INT PRIMARY KEY,
    name VARCHAR(255),
    nationality VARCHAR(100),
    birth_date DATE,
    death_date DATE
);
```

Normalization:

Normalization is a crucial step in database design, ensuring that data redundancy is minimized and data integrity is maintained. By organizing data into separate tables and eliminating redundant information, normalization up to the third normal form (3NF) ensures efficient data storage and retrieval, as well as facilitates easier maintenance and updates.

Entity-Relationship Diagram (ERD):

The ERD provides a visual representation of the relationships between entities in the BookWise database. It illustrates how different tables are connected through primary and foreign key constraints, highlighting the relational structure of the database and aiding in understanding its underlying architecture.



SQL Queries and Functionality

To cater to diverse informational needs, a suite of SQL queries has been developed, ensuring efficient extraction of data from the database. Here are examples exemplifying common queries and operations:

1. **Retrieve Fiction Books Details:**

   This SQL query retrieves the titles and authors of books belonging to the genre "Fiction".

```
SELECT title, author FROM Books WHERE genre_id = (SELECT genre_id FROM
Genre WHERE name = 'Fiction');
```

2. **Customer Details for "Norwegian Wood" Book Purchase:**

   This SQL query retrieves the names and email addresses of customers who purchased the book titled "Norwegian Wood".

```
SELECT name, email FROM Customers WHERE customer_id IN (SELECT customer_id
FROM Sales WHERE book_id = (SELECT book_id FROM Books WHERE title =
'Norwegian Wood'));
```

3. **Add Description Column to Genre Table:**

   This SQL statement adds a new column named "description" to the "Genre" table.

```
ALTER TABLE Genre ADD COLUMN description VARCHAR(255);
```

4. **Update Price of Haruki Murakami Books:**

   This SQL statement updates the price of books written by Haruki Murakami to $15.99.

```
UPDATE Books SET price = 15.99 WHERE author = 'Haruki Murakami';
```

5. **Retrieve IDs of Expensive or Available Books:**

   This SQL query retrieves the IDs of books that are available in inventory or have been sold for a price greater than $1000.

```
SELECT book_id FROM Inventory WHERE available = true UNION SELECT book_id
FROM Sales WHERE total_price > 1000;
```

6. Query to Get Total Sales by Employee:

This SQL query calculates the total sales made by each employee.

```
WITH TotalSales AS (
      SELECT employee_id, SUM(total_price) AS total_sales
      FROM Sales
      GROUP BY employee_id
)
SELECT e.name AS employee_name, ts.total_sales
FROM TotalSales ts
JOIN Employees e ON ts.employee_id = e.employee_id;
```

7. Retrieve Employee Details Using %ROWTYPE:

This PL/SQL block declares a record variable `emp_row` of type `Employees%ROWTYPE` and fetches details of an employee with `employee_id = 7` into this record.

```
SET SERVEROUTPUT ON;
DECLARE
      emp_row Employees%ROWTYPE;
BEGIN
      SELECT employee_id, name, email, phone_number, position, hire_date
      INTO emp_row.employee_id, emp_row.name, emp_row.email,
emp_row.phone_number, emp_row.position, emp_row.hire_date
      FROM Employees
      WHERE employee_id = 7;
END;
/
```

8. Retrieve Book Details Using Cursor and %ROWTYPE:

This PL/SQL block declares a cursor `book_cursor` to fetch all rows from the `Books` table and iterates through each row using a loop, printing details of each book.

```
SET SERVEROUTPUT ON;
```

```
DECLARE
      CURSOR book_cursor IS SELECT * FROM Books;
      book_row Books%ROWTYPE;
BEGIN
      OPEN book_cursor;
      FETCH book_cursor INTO book_row.book_id, book_row.title,
book_row.author, book_row.genre_id, book_row.price,
book_row.publication_date;
      WHILE book_cursor%FOUND LOOP
      DBMS_OUTPUT.PUT_LINE('Book ID: ' || book_row.book_id || ', Title: '
|| book_row.title || ', Author: ' || book_row.author || ', Genre ID: ' ||
book_row.genre_id || ', Price: ' || book_row.price || ', Publication Date:
' || book_row.publication_date);
      DBMS_OUTPUT.PUT_LINE('Row count: ' || book_cursor%ROWCOUNT);
      FETCH book_cursor INTO book_row.book_id, book_row.title,
book_row.author, book_row.genre_id, book_row.price,
book_row.publication_date;
      END LOOP;
      CLOSE book_cursor;
END;
/
```

9. SQL Function to Retrieve Customer Name:

This SQL code defines a function named `get_customer_name` that retrieves the name of a customer based on the provided `customer_id`. It then demonstrates how to use this function in a PL/SQL block to retrieve and print the name of a specific customer.

```
CREATE OR REPLACE FUNCTION get_customer_name(customer_id_param IN NUMBER)
RETURN VARCHAR2 AS
  customer_name_value Customers.name%TYPE;
BEGIN
  SELECT name INTO customer_name_value FROM Customers WHERE customer_id =
customer_id_param;
  RETURN customer_name_value;
END;
```

```
SET SERVEROUTPUT ON;

DECLARE
  customer_name_var VARCHAR2(255);
BEGIN
  customer_name_var := get_customer_name(201);
  DBMS_OUTPUT.PUT_LINE('Customer Name: ' || customer_name_var);
END;
/
```

10. Create Trigger to Update Inventory After Sale:

This SQL statement creates a trigger named `update_inventory_after_sale` that fires after an insertion on the `Sales` table. It updates the quantity in the `Inventory` table, decrementing it by 1, for the book that was sold.

```
CREATE OR REPLACE TRIGGER update_inventory_after_sale
AFTER INSERT ON Sales
FOR EACH ROW
DECLARE
      v_book_id Books.book_id%TYPE;
BEGIN
      v_book_id := :NEW.book_id;
      UPDATE Inventory
      SET quantity = quantity - 1
      WHERE book_id = v_book_id;
END;
/
```

Intended Users :

- **Book Aficionados**: The primary audience of the Bookstore Management System comprises individuals passionate about books, who seek access to a wide array of literary works, authors, and genres. This system caters to their needs by providing a user-friendly platform for browsing and discovering new titles..
- **Bookshop Owners and Managers**: Bookstore proprietors and managers rely on the Bookstore Management System to streamline their operations. From inventory management to sales tracking, the system offers robust features to efficiently manage bookstores, ensuring optimal stock levels and smooth transactions.
- **Literary Critics and Scholars**: Professionals engaged in literary analysis and academic research utilize the Bookstore Management System to access comprehensive book data. By leveraging the system's extensive database, they can delve into literary trends, analyze authorial styles, and conduct scholarly investigations.
- **Educators and Students**: Teachers and students in literature-related fields leverage the Bookstore Management System for educational purposes. The system's wealth of literary resources facilitates teaching and learning activities, allowing educators to create engaging lesson plans and students to explore diverse literary works.
- **Publishing Industry Professionals**: Professionals working in publishing firms find value in the Bookstore Management System for market analysis and sales tracking. By harnessing the system's analytics tools, they can monitor book sales trends, identify popular titles, and make informed decisions regarding publishing strategies.
- **Software Developers**: Developers specializing in book-related applications integrate the Bookstore Management System to enhance their products' functionality. Through integration with the system's API, they can offer features such as book recommendations, user reviews, and real-time inventory updates, enriching the user experience of their applications.

Conclusion :

- ## Reflection on Milestones and Obstacles :

  The journey of developing the Bookstore Management System has been marked by significant milestones and challenges. While overcoming complex database functionalities and implementing advanced features were notable achievements, challenges such as ensuring data integrity and system scalability required innovative solutions and performance optimization techniques.

- ## Importance of the System in the Literary Sphere :

  The Bookstore Management System holds substantial importance in the literary realm, serving as a vital tool for book enthusiasts, bookstore owners, academics, and industry professionals alike. By providing a reliable platform for book exploration, inventory management, scholarly research, and market analysis, the system contributes to the enrichment and advancement of the literary community.