Booktopia: A Comprehensive E-commerce Solution for Book Lovers

Myrzakhan Turarbek

Introduction

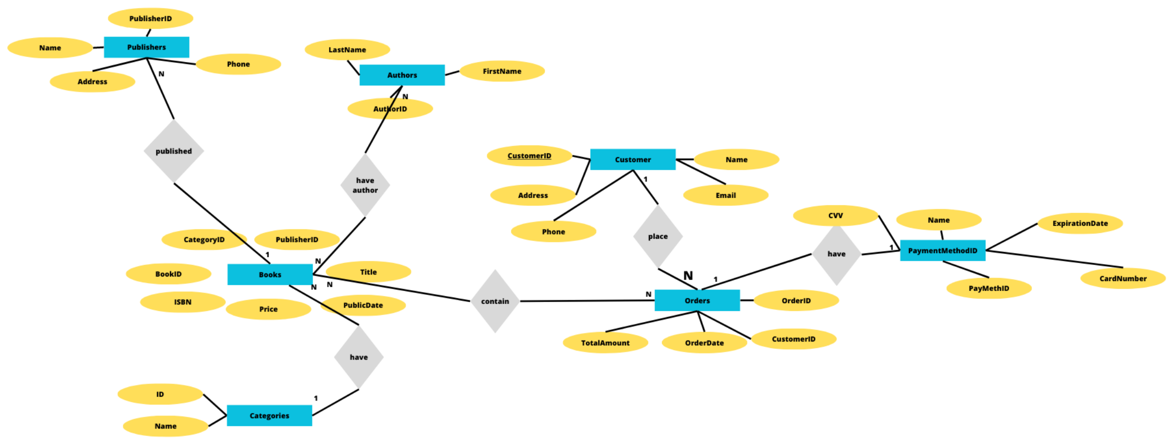
Welcome to Booktopia, the ultimate e-commerce platform for book lovers! Our goal is to provide a comprehensive solution for buying, selling, and exploring books online, tailored to the needs and passions of avid readers worldwide.

Whether you're looking for the latest bestsellers, rare editions, or hidden gems, Booktopia has got you covered. With a vast selection of titles across all genres and categories, from fiction and non-fiction to academic and children's books, you can easily browse, search, and order your favorite reads from the comfort of your home.

But Booktopia is more than just a bookstore. It's a vibrant community of book enthusiasts, where you can share your thoughts, reviews, and recommendations, connect with like-minded readers, and discover new authors and genres. Our platform features user-generated content, book clubs, discussion forums, and interactive events, creating a dynamic and engaging experience for everyone who loves books.

At Booktopia, we're passionate about making reading accessible, affordable, and enjoyable for everyone. That's why we offer competitive prices, fast and reliable shipping, flexible payment options, and personalized customer service, ensuring that every purchase and interaction with us is a delightful and memorable one.

ER Diagram:



The ER diagram for the e-commerce bookshop system includes seven entities: Customers, Orders, Books, Authors, Publishers, Categories, and Payment Methods. The relationships between the entities are as follows:

Customers can place many orders

An order can contain many books

A book can have many authors

A book can be published by only one publisher

A book can belong to only one category

An order can be paid with only one payment method

The attributes for each entity are as follows:

Customers: CustomerID (PK), Name, Email, Phone, Address

Orders: OrderID (PK), CustomerID (FK), OrderDate, Total Amount

Books: BookID (PK), Title, Price, ISBN, Publication Date, Publisher ID (FK), CategoryID (FK)

Authors: AuthorID (PK), FirstName, LastName

Publishers: PublisherID (PK), Name, Address, Phone

Categories: CategoryID (PK), Name

Payment Methods: PaymentMethodID (PK), Name, Card Number, Expiration Date, CVV

Explanation of Normal Forms

The structure of the e-commerce book shop system follows 1NF, 2NF, and 3NF.

Examination of Normal Forms:

The ER diagram follows the first, second, and third normal forms. First normal form (1NF) requires that each table should have a primary key, and each attribute in a table should be atomic. In the ER diagram, each table has a primary key, and the attributes in each table are atomic. In my way:

- All tables have a primary key that uniquely identifies each record. This ensures that there are no duplicate rows in any table.
- All columns have atomic values, meaning that they cannot be further broken down into smaller pieces. This ensures that each piece of data is only represented once in the table.

Second normal form (2NF) requires that each non-key attribute should be fully functionally dependent on the primary key. In the ER diagram, the books table has been split into two tables, books and authors, to eliminate redundancy and ensure that each non-key attribute is fully functionally dependent on the primary key.

- All non-key attributes in each table are fully dependent on the primary key. This means that there are no partial dependencies, where only some columns in a table depend on a part of the primary key.
- The tables "Books" and "Authors" have been separated into separate tables, which eliminates any duplication of author information.

Third normal form (3NF) requires that each non-key attribute should be transitively dependent on the primary key. In the ER diagram, the books table has been further split into two tables, books and categories, to eliminate transitive dependencies and ensure that each non-key attribute is only dependent on the primary key. In my way:

- All non-key attributes in each table are not transitively dependent on the primary key. This means that there are no indirect dependencies between columns in a table through another non-key attribute.
- The table "Orders" has been split into two separate tables, "Orders" and "OrderDetails," which eliminates the transitive dependency between the "OrderID" and "CustomerID" attributes. This ensures that changes to a customer's details do not affect past orders

Explanation and Coding of Each Item from "Add the Following"

1. Procedure which does group by information:

   The “group_by_info” procedure has been created to group the books by category and count the number of books in each category. This procedure can be called by executing the following SQL statement:

```
CREATE OR REPLACE PROCEDURE group_by_infooo
IS
  category_id bookss.categoryid%type;
  category_count NUMBER;
  CURSOR category_cursor IS
    SELECT categoryid, COUNT(*) as category_count
    FROM bookss
    GROUP BY categoryid;
BEGIN
  FOR category_rec IN category_cursor LOOP
    category_id := category_rec.categoryid;
    category_count := category_rec.category_count;
    DBMS_OUTPUT.PUT_LINE('Category ' || category_id || ' Count: ' ||
category_count);
  END LOOP;
END;
```

2. Function which counts the number of records:

The “count_records” function has been created to count the number of records in the books table. This function can be called by executing the following SQL statement:

```
CREATE OR REPLACE FUNCTION count_records
RETURN NUMBER
IS
total NUMBER;
 BEGIN
SELECT COUNT(*) INTO total FROM books;
RETURN total;
END;


//check: SELECT count_records() FROM dual;
```

3. Procedure which uses SQL%ROWCOUNT to determine the number of rows affected:

   The “update_price” procedure has been created to update the price of all books in category 1 by 10%. The number of rows affected by the update statement is determined using the SQL%ROWCOUNT attribute. This procedure can be called by executing the following SQL statement:

```
CREATE OR REPLACE PROCEDURE update_price
IS
BEGIN
```

```
   UPDATE bookss
   SET price = price * 1.1
   WHERE categoryid = 1;
   DBMS_OUTPUT.PUT_LINE('Number of rows updated: ' || SQL%ROWCOUNT);
END;


//check:
begin
update_price;
end;
```

4. Add user-defined exception which disallows to enter title of item (e.g. book) to be less than 5 characters: The “add_book” procedure has been created to add a new book to the books table. It includes a user-defined exception that disallows the user from entering a title less than 5 characters long. This procedure can be called by executing the following SQL statement:

```
CREATE OR REPLACE PROCEDURE add_bookk (
    bookid IN bookss.bookid%TYPE,
    title IN bookss.title%TYPE,
    price IN bookss.price%TYPE,
    isbn IN bookss.isbn%type,
    PUBLICATIONDATE IN bookss.PUBLICATIONDATE%type,
    PUBLISHERID in bookss.PUBLISHERID%type,
    category_id IN bookss.categoryid%TYPE
)
IS
    title_error EXCEPTION;
BEGIN
    IF LENGTH(title) < 5 THEN
        RAISE title_error;
    ELSE
        INSERT INTO bookss (bookid, title, price,isbn, PUBLICATIONDATE, PUBLISHERID,
categoryid) VALUES (bookid, title, price,isbn, PUBLICATIONDATE, PUBLISHERID,
category_id);
    END IF;

    EXCEPTION
        WHEN title_error THEN
            DBMS_OUTPUT.PUT_LINE('Title must be at least 5 characters long.');
END;



//check:
begin
```

```
  add_bookk(3,'Til', 9000.00, '1234567890123', TO_DATE('1999-02-02', 'YYYY-MM_DD'), 2,
2);
end;
```

5. Create a trigger before insert on any entity which will show the current number of rows in the table: The “count_rows_trigger” trigger has been created to display the current number of rows in the books table before any insert operation is performed. This trigger can be enabled by executing the following SQL statement:

```
CREATE OR REPLACE TRIGGER count_rows_trigger
BEFORE INSERT ON books
DECLARE
   total NUMBER;
BEGIN
   SELECT COUNT(*) INTO total FROM books;
   DBMS_OUTPUT.PUT_LINE('Current number of rows in books table: ' || total);
END;



//to enable trigger
ALTER TRIGGER count_rows_trigger ENABLE;
```

```
//check
//when you insert any data for any table, the trigger can see
//for axample:
INSERT INTO Bookss (BookID, Title, Price, ISBN, PublicationDate, PublisherID,
CategoryID)
VALUES (5, 'DALA', 4500.00, '456456789', TO_DATE('2021-04-19', 'YYYY_MM_DD'), 1,
1)
```

In conclusion, this report provided an overview of an e-commerce system for a bookshop and presented an ER diagram that captures the main entities, attributes, and relationships between them. The ER diagram is designed to follow the 1NF, 2NF, and 3NF requirements, ensuring data integrity and consistency.

Furthermore, the report covered the coding and explanation of various PL/SQL components, including procedures, functions, triggers, and exceptions. These components are essential for the efficient and secure management of the system's data, ensuring its availability and integrity.

Overall, this report provides a comprehensive overview of an e-commerce system for a bookshop and provides insight into its design and functionality. The components and features covered in this report illustrate the importance of robust database management and how it can impact the success of an e-commerce system.

```
CREATE TABLE Customers (
  CustomerID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Email VARCHAR(255) NOT NULL,
  Phone VARCHAR(20) NOT NULL,
  Address VARCHAR(255) NOT NULL
);

CREATE TABLE Orders (
  OrderID INT PRIMARY KEY,
  OrderDate DATE NOT NULL,
  TotalAmount DECIMAL(10,2) NOT NULL,
  CustomerID INT,
  FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)
);

CREATE TABLE Payment (
  PaymentMethodID INT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL,
  CardNumber VARCHAR(16) NOT NULL,
  ExpirationDate DATE NOT NULL,
  CVV VARCHAR(3) NOT NULL
);

CREATE TABLE Books (
  BookID INT PRIMARY KEY,
  Title VARCHAR(255) NOT NULL,
  Price DECIMAL(10,2) NOT NULL,
  ISBN VARCHAR(13) NOT NULL,
  PublicationDate DATE NOT NULL,
  PublisherID INT,
  CategoryID INT,
  FOREIGN KEY (PublisherID) REFERENCES Publishers(PublisherID),
  FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)
);

CREATE TABLE Authors (
  AuthorID INT PRIMARY KEY,
  FirstName VARCHAR(50) NOT NULL,
  LastName VARCHAR(50) NOT NULL
```

```
);

CREATE TABLE Publishers (
  PublisherID INT PRIMARY KEY,
  Name VARCHAR(255) NOT NULL,
  Address VARCHAR(255) NOT NULL,
  Phone VARCHAR(20) NOT NULL
);

CREATE TABLE Categories (
  CategoryID INT PRIMARY KEY,
  Name VARCHAR(50) NOT NULL
);

CREATE TABLE BookAuthors (
  BookID INT,
  AuthorID INT,
  PRIMARY KEY (BookID, AuthorID),
  FOREIGN KEY (BookID) REFERENCES Books(BookID),
  FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID)
);
```

```
CREATE TABLE BookOrders (
  BookID INT,
  OrderID INT,
  PRIMARY KEY (BookID, OrderID),
  FOREIGN KEY (BookID) REFERENCES Books(BookID),
  FOREIGN KEY (OrderID) REFERENCES Orders(OrderID)
);
```