The Internet BookShop
DBDudes, Inc.
Computer Scientists:

- Allen Lee
- Robert Smith

# I. Overview

Our group, DBDudes, consisting of members Robert Smith and Allen Lee, proposes creating a new Internet bookstore called *The Internet BookShop*. The purpose of our website is to create a down to earth environment where Internet users from around the world can purchase their favorite books.

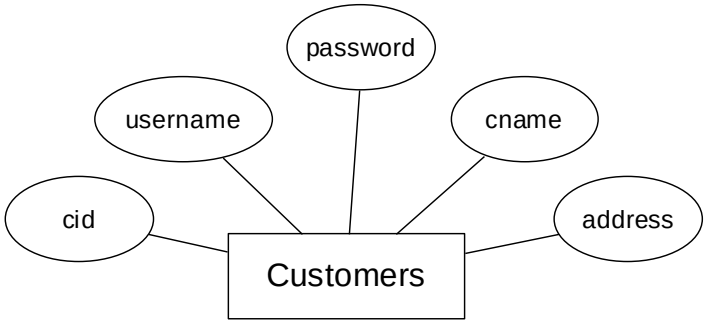
The website will be similar in structure to Amazon.com, the current Internet bookstore leader, but we hope to provide a superior user experience along with better pricing. In addition, we plan to utilize modern internet technologies such as JDBC, cookies, and servlets in our project.

# II. Database Design

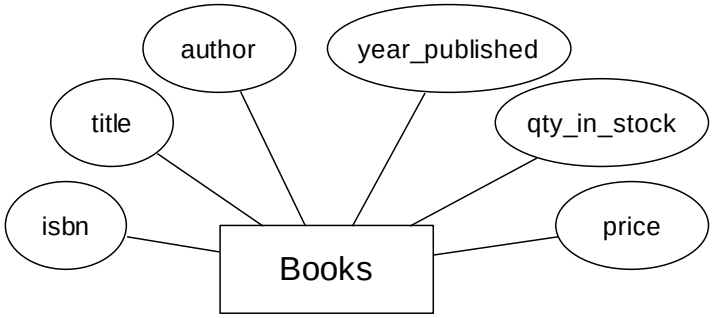
The following diagrams provide a high-level introduction to our database. They are provided as reference while learning about the system's client functionality in the following section.

**Entity-Relationship Diagrams:**

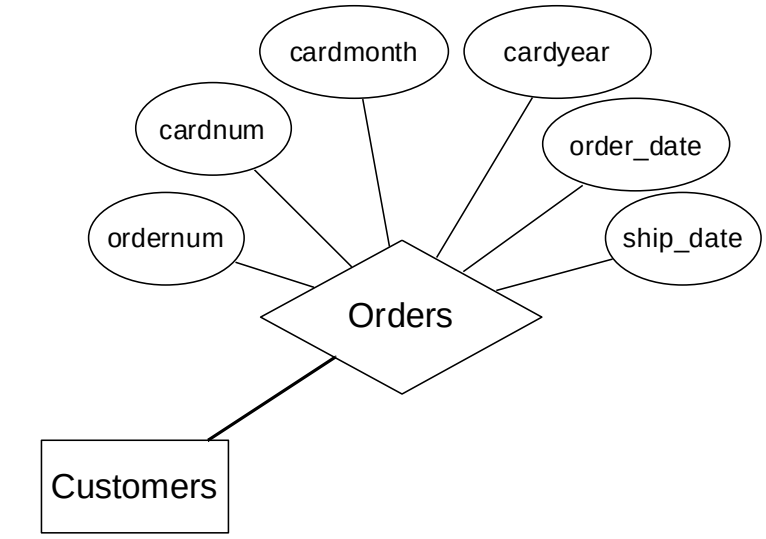
**Customers**

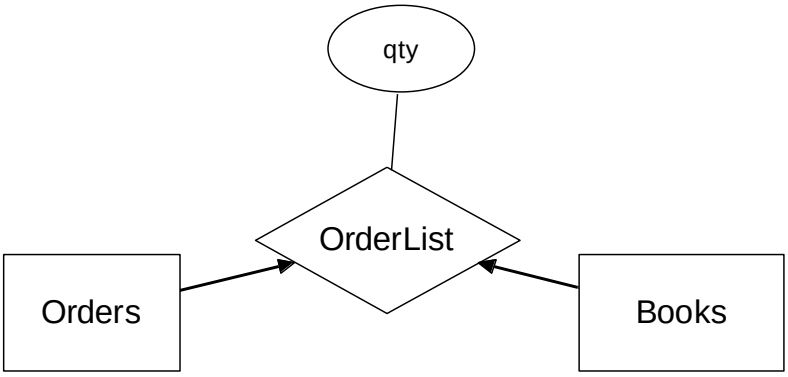



**Books**



**Orders**


cardmonth
cardyear
cardnum
order_date
ordernum
ship_date
Orders
Customers


qty
OrderList
Orders
Books

# III. Client Functionality

This section describes how our user interface will work and how customers will view and be able to interact with our web site. A more detailed description of how different pages work internally will be discussed in following sections.

We start by going over the types of users on our system. There are two different classes:

1. Anonymous Users

   All users who initially connect to our website are in this class and maintain in it until they log in with an account. They can search for books online, view book information for specific books, add books to their shopping carts, modify books in their shopping cart, view help menus, create new user accounts, and login to the system.

2. Logged in Users

   Once users have logged in from a previously created account, they have all the rights and functionality of an anonymous user as well as the ability to proceed to the checkout. Once there, they can confirm and then place their order.

We will now examine the experience that each class of user will have during a visit to our website.

The anonymous user can view 10 types of pages:

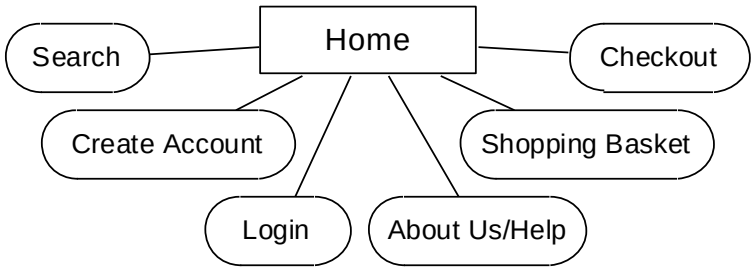
1. Home Page
2. Search Page
3. Book Information Page
4. Account Creation Page
5. Shopping Basket
6. Login Page
7. Help/Information Page
8. F.A.Q.
9. System Rules Page
10. Contact Page

The logged in user can view also view:

11. Checkout Page

**Common to All Page**

Every page will have a menu bar at the top. This will allow users to move fluidly around our system from any page. All users, whether logged in or not, will have access to the same menu bar although some of the web pages may not be available unless a user has logged in. The menu bar will offer the following options:




***The Anonymous User***

**Page 1: Home Page**

The home page will be the first page users see when they connect to the website. Users will be greeted by a welcome message for our web site. They will also be presented with the following list of options:

| *Action* | *Directed To* |
|---|---|
| Create new Account | **Page 4** |
| Login | **Page 6** |

The application layer will display the same HTML home page for all users who connect to the system.

**Page 2: Search Page**

This page will consist of a series of questions that will assist a user in searching our database. A User selects whether they would like to search for a

book by author, title, or ISBN number. They then enter a non-case sensitive text string used to search for their book. Some examples of possible queries are:

Author = Charles Dickens
Title = American Pastoral
ISBN = 1829853333

Users will be presented with a list of books and have the ability to view individual books book information pages, **Page 3**, for each book in the database.

The application layer will load a generic web form as described above. When the user submits the query, the application layer will call a JSP to retrieve the requested information from the database as well as present this information to the user as HTML.

### Page 3: Book Information Page

For each book on the website, the server will generate a unique book information page based on the ISBN number of the book. Users will be presented with information such as the books title, author, price, and availability.

Based on the quantity of the book currently in stock, the user will receive one of the following messages about the book's availability:

| Quantity in Stock | Availability Message Displayed |
|---|---|
| < 0 | Usually ships within 4 weeks |
| 0-4 | Usually ships within 1 week |
| 5-19 | Usually ships within 2 to 3 days |
| ≥ 20 | Usually ships within 1 to 2 days |

The application layer will generate a unique book information page by querying the database with the provided ISBN number in the link. For example, a call for a book with the ISBN number of '123222111' would be of the following form:

http://server.com/bookhome.jsp?ISBN=1232221111

The application layer will also check for invalid, missing, or unknown ISBN numbers and report to the user if the book they requested cannot be found on the system.

**Page 4: Account Creation Page**

Users without accounts or those wishing to create new ones can do so by choosing to create a new account from the common menu bar or from the system home page, **Page 1**. They will be asked a series of personal questions including choosing a new username for the system as well as a password. If any of the information they provide is incorrect, missing, or invalid, they will be asked to enter it again. For example, if the user enters a password that is too short or choosing a username that is all ready in the system, they will be informed that these fields are invalid and that they must re-enter their information before proceeding.

After the account information has been entered correctly, the account will be created by the system and the user will be automatically logged in and directed to their shopping basket, **Page 5**.

The application layer will check the information the user enters thoroughly each time they submit it before attempting to create the account. If all the data passes, the server will attempt to make the account and check for account creation errors.

In addition, the Account Creation Page will invoke the Login Page's JSP file from **Page 6** in order to log the user into their account directly after it has been created.

**Page 5: Shopping Basket**

The shopping basket is a temporary state page that shows the user which items they have added to their order during their current visit to the website. In addition, users can change their shopping cart by both modifying the quantities of books in their baskets, as well as deleting books from their basket. To make changes to their shopping cart, users will need to click an **Update** button that will reload their shopping cart with their new options present.

Note that no information contained in the shopping basket is maintained by the system permanently; so that a user's shopping basket is erased each time they connect to our server.

The application layer will present a form to the user listing each of their books and the current quantities of each book they wish to order. The form will allow users to modify these quantities taking care to allow only valid

integers from a valid numerical range. The form will submit to itself by saving the shopping cart information to a cookie on the user's computer and then reloading the shopping cart based on the user's new cookie information.

**Page 6: Login Page**

The login page will ask a user for their username and password to use to log into the system. In addition, they will have the following options based on the information they provided:

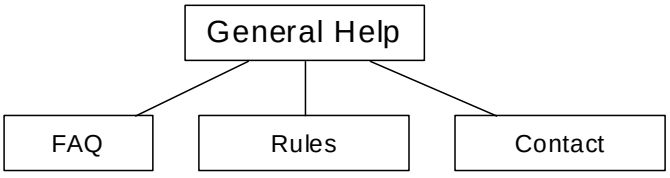
| *Action* | *Directed To* |
|---|---|
| Forgot Password | **E-mail Message to Administrator** |
| Login: Incorrect Information | **Page 5** |
| Login: Correct Information | **Back to Page 6** |

The application layer will query the database with the information the user has provided and if it is valid for a user in the system, it will log a user in by creating a cookie on the user's machine.

**Page 7: Help/Information Page**

The help/information Page is a sub-section of the main website, which attempts to help users who may have difficulty, find their way. They are greeted by a welcome message telling them about our site as well as given the choice to view the following pages:

| *Action* | *Directed To* |
|---|---|
| View F.A.Q. | **Page 8** |
| View Rules | **Page 9** |
| Display Contact Information | **Page 10** |




On all help pages, the application layer will simply load HTML webpages for any user requesting the information.

**Page 8: F.A.Q.**

The F.A.Q. Page, or Frequently Asked Questions Page, will inform the user of common questions many users have about the system or how to use the system as well as providing a solution to the problem.

**Page 9: System Rules Page**

Our website will have a page displaying system rules that users, by using our system, are required to follow.

**Page 10: Contact Page**

This page will make available to the user information on how to contact a staff member via e-mail to help them with any problem they might have.

***The Logged In User***

**Page 11: Checkout Page**

Only when a user has at least one item in their shopping basket and has logged into our system will they be able to proceed to the checkout page. First, they will be asked to confirm the items in their order. Next, they will be asked to provide a valid credit card for the order as well as shown their shipping address for the order. Finally, they will be shown the information for their entire order and asked to confirm it. Once they confirm the order, it is sent to the database for processing.

Even if quantities of a specific book are not available at the time of purchase, the transaction will still be completed. If a book's quantity in stock falls to below zero, that will represent to the store owners that X number of holds have been placed on the book, to be filled as soon as new copies of the book arrives.

The application layer will present multiple forms to the user each asking them a small set of questions including asking them for their credit card information and whether they would like to confirm their order or not. Each form will submit to a JSP, which will then generate a new form based on the results of the previous form. When all the information has been processed

and confirmed, the order will be added to the database and the user will be informed.

In addition, the quantities of the books in stock will be decremented by the quantities of books the user purchased for each book.

# IV. Database Tables

This section describes what data we will store in our database as well as how our relations represent this data.

The following tables will be included in our database. Please refer to the E-R diagrams at the beginning of the document for a higher-level graphical description.

**Customers**

Customers(*cid*: INTEGER, *cname*: CHAR(80), *address*: CHAR(200),
*username*: CHAR(16), *password*: CHAR(16))

**foreign keys:** *none*
**candidate key:** *username*
**primary key:** *cid*
**not null:** *cid, cname, address, username, password*

This table stores all of the information associated with a given user. It is used to authenticate users attempting to login as well as retrieving shipping information about user during check out. A user's *cid* is used by other tables to track a user's orders and movement in our system.

**Books**

Books(*isbn*: CHAR(10), *title*: CHAR(80), *author*: CHAR(80),
*qty_in_stock*: INTEGER, *price*: REAL, *year_published*: INTEGER)

**foreign keys:** *none*
**primary key:** *isbn*
**not null:** *isbn, title, author, qty_in_stock, price, year_published*

Each entry in this table represents a specific book the store carries. When a user searches the database for a specific book, this table is consulted for that information. When a user requests information about a specific book, we present publishing information including ISBN, title, author, and year as well as our system information including our price and quantity in stock.

We use ISBN numbers as the primary key because this system is acknowledged worldwide and no two books ever have the same ISBN number.

**Orders**

Orders(*ordernum*: INTEGER, *cid*: INTEGER, *cardnum*: CHAR(16),
*cardmonth*: INTEGER, *cardyear*: INTEGER, *order_date*: DATE,
*ship_date*: DATE)

**foreign keys:** *cid (Customers.cid)*
**primary key:** *ordernum*
**not null:** *ordernum, cid, cardnum, cardmonth, cardyear*

This table stores one entry for each order the user places. It only includes general information about the transaction itself, and not the specific items purchased by the user. The *cid* of the user placing the order as well as their credit card number and credit card date are stored here. In addition, we can track when the order was submitted as well as when it shipped.

**OrderList**

OrderList(*ordernum*: INTEGER, *isbn*: CHAR(10), qty: INTEGER*)*

**foreign keys:** *ordernum (Orders.ordernum), isbn (Books.isbn)*
**primary key:** *(ordernum,isbn)*
**not null:** *ordernum, isbn, qty*

This table records the items purchased for each transaction in the Orders table as well as their associated quantities.

For example, if a user purchased two copies of one book and one copy of another book, then there would be a single entry in the Orders table and two entries in the OrderList table. The two copies of the first book would compose the first entry in the OrderList table while the single copy of the second book would compose the second entry.

# V. SQL Queries

### Search Queries

When a user queries the database using our search page, he can search by author, title, or ISBN number using a specified search string, `<SearchString>`, and the results are returned in alphabetical order based on title. This is implemented by one of the following three queries:

```
SELECT isbn, title, author, price
FROM Books
WHERE author LIKE '%<SearchString>%'
ORDER BY title

SELECT isbn, title, author, price
FROM Books
WHERE title LIKE '%<SearchString>%'
ORDER BY title

SELECT isbn, title, author, price
FROM Books
WHERE isbn LIKE '%<SearchString>%'
ORDER BY title
```

The queries are all so similar that they can be built from the same string with simply one word, the column to search on, modified. If the `<SearchString>` is left blank, it will signify a wildcard search in the database, retrieving all books available. In addition, text matches will be formatted so that they are case insensitive during a search.

### Book Information Page

Information for each book will be retrieved solely based on the ISBN number provided, `<SpecifiedISBN>`, and the following query:

```
SELECT title, author, qty_in_stock, price, year_published
FROM Books
WHERE ISBN = '<SpecifiedISBN>'
```

Because the ISBN number is a primary key on **book**, and since we are using an equality match rather than a `LIKE` match as we did to search for sets of books, we are guaranteed that at most one book will be returned by this query.

### Create New Account

During the process of creating a new account, the server will verify that the username desired by the user does not match a username all ready in the database using the following query:

```
SELECT username, password
FROM Customers
WHERE username = '<Username>'
```

If so, the user is told to select a new username. The application server compares all usernames in a case insensitive manner, so that no two usernames formatted with different cases can be created. After this process is complete, the server creates the new account using the information provided by the user and the following query:

```
INSERT INTO Customers (cname,address,username,password)
VALUES ('<CustomerName>','<CustomerAddress>',
        '<Username>','<Password>')
```

If there are any database problems during the creation of the account, most likely due to concurrent access or the server being unavailable, the user is informed and the creation process is aborted.

### Shopping Basket

For each book in the user's shopping basket, the system will retrieve its ISBN number, title, and price and display it to the user:

```
SELECT isbn, title, price
FROM Books
WHERE ISBN = '<SpecifiedISBN>'
```

As with the query used for the book information page, we are guaranteed at most one result per query.

### Login Page

The login page takes a given username and searches for a matching record in the Customers table using the following query:

```
SELECT cid, username, password
FROM Customers
WHERE username = '<SpecifiedUsername>'
```

Again, the username is formatted so that the match is case insensitive. If an account is found matching the `<SpecifiedUsername>`, the password is then examined and checked against the one provided by the user. Unlike username checks, password checks are case sensitive.

**Checkout**

Page 1 – Verify Order: As with the shopping basket page, the system retrieves the ISBN number, title, and price for each book in the user's shopping basket and displays it to the user:

```
SELECT isbn, title, price
FROM Books
WHERE ISBN = '<SpecifiedISBN>'
```

Page 2 – Verify Shipping Information: Using the login cookie created at the time of login, the system retrieves the user's <Cid> and then queries the database using this primary key to retrieve the user's shipping address.

```
SELECT cname, address
FROM Customers
WHERE cid = <Cid>
```

Page 3 – Confirm Order: On the final page before submitting the user's order, we present their shopping basket one last time using and use the following query to retrieve information on each book the user is ordering:

```
SELECT title, price
FROM Books
WHERE ISBN = '<SpecifiedISBN>'
```

Page 4 – Submit Order: Once all the information has been verified, we begin processing the transaction. If there are any errors during any point in the transaction, we rollback the entire transaction and inform the user of an error.

The first step is to create a single unique entry in the Orders table:

```
INSERT INTO Orders (cid,cardnum,cardmonth,cardyear)
VALUES (<Cid>,'<CreditCardNumber>',<CardMonth>,<CardYear>)
```

Afterwards, we need to query the database to retrieve the new order number created by the previous SQL statement:

```
SELECT ordernum
FROM Orders
WHERE CID = <Cid>
ORDER BY ordernum DESC
```

Note that the JDBC interface offers a more eloquent way of doing this, by returning any new order numbers generated by original insert statement, but this option is currently not fully functional in the most recent Java release.

Finally, for each unique book in the transaction, we add an entry into the OrderList table containing the order information, book information, and number of quantities ordered.

```
INSERT INTO OrderList (ordernum,isbn,qty)
VALUES (<OrderNumber>,<ISBN>,<cid>)
```

# VI. Database Initialization Queries

The following four SQL queries are used to initialize the database tables:

```
CREATE TABLE Customers
       (cid INTEGER NOT NULL generated always
          AS IDENTITY (start with 0, INCREMENT by 1, NO CACHE)
          PRIMARY KEY,
        cname CHAR(80) NOT NULL,
        address CHAR(200),
        username CHAR(16) NOT NULL, UNIQUE (username),
        password CHAR(16) NOT NULL)

CREATE TABLE Books
       (isbn CHAR(10) NOT NULL PRIMARY KEY (isbn),
        title CHAR(80) NOT NULL,
        author CHAR(80) NOT NULL,
        qty_in_stock INTEGER NOT NULL,
        price REAL NOT NULL,
        year_published INTEGER)

CREATE TABLE Orders
       (ordernum INTEGER NOT NULL generated always
          AS IDENTITY (start with 0, INCREMENT by 1, NO CACHE)
          PRIMARY KEY,
        cid INTEGER NOT NULL,
        cardnum CHAR(16) NOT NULL,
        cardmonth INTEGER NOT NULL,
        cardyear INTEGER NOT NULL,
        order_date DATE,
        ship_date DATE,
        FOREIGN KEY (cid) REFERENCES Customers ON DELETE CASCADE)

CREATE TABLE OrderList
       (ordernum INTEGER NOT NULL,
        isbn CHAR(10) NOT NULL,
        PRIMARY KEY (ordernum,isbn),
        qty INTEGER NOT NULL,
        FOREIGN KEY (ordernum) REFERENCES Orders ON DELETE CASCADE,
        FOREIGN KEY (isbn) REFERENCES Books ON DELETE CASCADE)
```

The following two SQL queries are used to initialize the database triggers:

```
// After an order is created, this trigger sets the order date
CREATE TRIGGER orders_update
      AFTER INSERT ON Orders
      REFERENCING NEW AS N
      FOR EACH ROW
      MODE DB2SQL
            UPDATE Orders
            SET order_date = CURRENT DATE
            WHERE ordernum=N.ordernum


// After each element of an order is added to the database,
// update the quantities available of that item
CREATE TRIGGER orderlist_update
      AFTER INSERT ON OrderList
      REFERENCING NEW AS N
      FOR EACH ROW
      MODE DB2SQL
            UPDATE Books
            SET qty_in_stock = ((SELECT qty_in_stock
                                 FROM Books B
                                 WHERE B.isbn=N.isbn)-N.qty)
            WHERE isbn=N.isbn
```