Bookstore Database Project Report

CMPSC431W

John Hofbauer

Table of Contents

List of Figures

1. Introduction

This document states and elaborates upon the ‘Bookstore’ database design for Penn State’s CMPSC431W class. The following outlines the development and design for part one; clarifying the requirements: tasks, requirement analysis, conceptual database design, technology survey, and logical database design and normalization.

1.1. Organization

This project is divided into three parts: User, administrator, system. Where the User will view a GUI (Graphical User Interface) from a website, while having access to the database through a development stack. The user will have access to a limited number of operations to interact with the environment. Provided that they have the qualification to login/register as a user. Managers however, will have increased privileges providing them with the ability to terminate user accounts, delete and remove non-applicable comments/ratings. This assignment demonstrates Database Independence (a property of DBMS), which allows the database schema to change without requiring the change of a schema at a higher level. This also segregates the data from all the programs that use the information stored. This is essential when trying to reach multiple users with a reliable and secure database. The result of which will withhold personally identifiable i.e.: SSN, passwords (stored in hash encryption), and address’ from users not authorized to access them and prevent security leaks since the data is managed by a DBMS.

2. Requirement Analysis

2.1. Customer Registration

New users (clients) must provide the required information: name, login name, password, address, and phone number. The user will input the required information into the specified field as designed in the layout in HTML and CSS on the registration page. The system will then verify the registering username through a database call making sure that the 'login name' is not taken. However, if needed the system will prompt the user of any identified issues and request a different login name. Once a submission is made with a unique username, the resulting fields are inserted into the database; where the login name is the Primary key and the password is saved in the converted hash form (to future proof in the event of a data leak). The data that is introduced is checked to be the correct type and there should be no information edited or removed in the process.

2.2. Multiple Managers

When hiring new managers; introducing them into the system will be done through a superuser/manager, or another co-worker. The manager has access to the database through the superuser (the employee account that is created by the system at the time the database is first deployed) to create new employees in the database by logging into the system as a verified employee. The new employee will then enter the required information: full name, login name, password, address, phone number to be registered. Afterwards, any employee can login and edit/remove/add a book entry. This will be done by requesting the book from the database and submitting a request to the system that is converted into an 'update' SQL command for the resulting book referring to the correct ISBN number.

Since the first account ever created on the database (root account) will have a default password, it is required to be changed before entering any new employees. the absence of doing so could result in the compromise of the system; where an unauthorized could guess and be granted unauthorized access into the system.

2.3. Ordering

A user/customer can order any number of books, including duplicates. The user must first login, then they may create a request/order for several quantities or books with a minimum amount of one. The customer may not order a book that is not held within the database (every ISBN number for each book with their resulting stock must be checked to produce and verify the availability). Thus, the books must be present, have the requested amount of stock for each book, and the order must not be empty. The system will process the user's request and check the availability/stock for every book checking that the customer requested less or equal to the stock stored. This will then output the purchase of every book and the count or an error expressing to the user that they have not ordered any books (the required entry fields are empty) or that there is not enough stock to process the order.

2.4. New books

Only employees may edit/remove/add a book entry and must login to be verified by the system before the server may send, update, delete, or insert queries to the DBMS. The employee will input their login name and password, that will be hashed and checked with the database as a verified user. The employee will enter the new book'(s) ISBN, Title, Author(s), publisher, language, publication date, number of pages, stock level, price, keywords, and subject. The system will convert into an SQL insert and the server will send a message to the DBMS to add the books. No indirect data should be affected.

2.5. Arrival of more copies

When adding more copies of the same book. An employee must login (with login name and password) first, and the system will verify them as an existing/authorized employee. Then they may update the stock level of a certain book. This is done through the ISBN number, where the employee may search for the desired book, verifying the ISBN and choosing one to update the stock level of. This may be done by changing the stock level explicitly or adding several books to the current count. No other information besides the stock number should be altered.

2.6. Comments

Customers may enter a comment for a book. The customer must login and select the ISBN to create a comment for. When choosing a book to make a comment on, the user may search for a book to find the ISBN to attach the comment to. However, the user may only make one comment for every book. Due to this structuring, every book the user searches for will have a resulting comment field attached to it. The user will see their one and only comment that they can add/remove/edit. The primary key for all comments will be a combination of the books ISBN and the user's login name, and a comment id as a foreign key. The comments will also be deleted whenever any of the primary keys are removed. (on update delete)

2.7. Usefulness rating

Customers may rate other's comments for a certain book. Comments will be presented under each book (for each ISBN). When the user is searching for a book, they may choose to read and rate the comments that other customers have left for a certain book useless/useful/very useful. Every rating within the rating table will have the comment id and the user's username as the primary key. A user may rate a comment without knowing the comment id and the usefulness will be updated to an integer scale to save data. If the user has already given a rating to this specific comment, then the new score is updated, and the user is notified by the bolding on the words that they have chosen. Only the rating for that comment – on that book – for the current user is ever created/changed/removed.

2.8. Trust Recording

Customers may untrust or trust any other users. Meaning that they can select any login name and add them to the specified list. The trust record will be a weak entity set for every customer (the primary key for the trusted and not-trusted table is the user's login name) so that they can have any number of trusted/untrusted users. The system will output a list of trusted and untrusted users that can be managed by the current customer. There will be a numerical value stored for each trust record, '-1' being untrusted and '1' representing trusting. This will update/add/remove trusted and untrusted users from the trust table and as a result will require the customer to log in.

2.9. Book Browsing

When a user searches for books, the user may use a combination of keywords to find specific books by author, publisher, title, language. These results will then be sorted by publish date, the average score for comments, or the score of trusted user's feedback through the use of a combo-box within the GUI. The user is not required to sign in to browse books however they will not be able to modify any user-specific data until they have done so. The server will convert the requested search into SQL for the DBMS before the system returns with the resulting books and their acquired comments. The database is not modified by search queries and therefore no information is changed when the user is not logged in.

2.10. Useful Comments

The customer can request (SQL equivalent being select) the top/most-useful N – where N is the number of comments – for a book. The useful score is calculated by the comment's usefulness score. The numeric ratings from *2.7 Trust Recording* are converted into an average in

SQL using the ‘group by’ for every comment and sorted by descending. The user will receive the resulting comments in order. No data will be updated or added as a result of this query since no login is required.

2.11. Buying Suggestions

Included with every order confirmation page (as a result of *2.3. Ordering*) will be a list of suggested books that have similarities – another customer who bought book A also bought book B – sorted in descending order by the number of customers who made a subset of the same order (Book A and B were in their order). The customer will input no more information that has already been input from *2.3. Ordering* and the system will output the list of suggestions. No data within the database will change because of this search. (SQL equivalent being select)

2.12. Degrees of Separation

A customer may choose to find books from authors who are 1-degree or 2-degree from a certain author. The user can select an author from a book or search for one. Then they may choose the degree of separation – combo-box of one or two – where the system will request to the server the desired information (author and degree of separation). The server will create a SQL query for books that are co-authored by the desired author for 1-degree; and books that are co-authored by two authors X and Y that are 1-degree from Z but (SQL equivalent being Intersect) X and Y are not 1-degree from each other: for 2-degree authors. This will return a list of the resulting books and will not affect any data within the database, since login is not required for searching for books *2.9 Browsing Books.*

2.13. Book Statistics

Every semester the manager will receive (1) a list of the most popular books, for a number 'N' (SQL equivalent being limit N) books within the current quarter (with a date greater than the quarter's start date). A second list (2) of the most popular authors for several authors 'N'. (group by authors sales / count sales / order descending / limit N) also within the current quarter. Finally, a list of the most popular publishers (group by publishers for sales/count sales/order descending / limit N) for the current quarter. These queries, like before, will not affect the data within the database and should only need to query within the orders table.

2.14. User Awards

The manager will give awards to the best users by logging in to verify they are the superuser. Then they will obtain the result by requesting the top 'N' most trusted users. This is calculated by grouping by login name, for the table of trusted users and summing the trusting score, since a trusting score is stored as a '1', and non-trusting is stored as a '-1' the addition of which will result in the trust score. The manager will also receive the 'N' most useful users using the same design as the trusting score. This will not change any information within the database but will require the manager to log in to access this information.

3. Additional Proposed Functionality

3.1. Available Login Names

When a user is creating a new account and the current login name is taken, it may result in many tries to guess a unique login name, therefore there will be a list of similar login names that are available (By putting numerical numbers at the end of the login name and checking if they exist). The user may choose from the list or continue to create a different unique login name. this is at the setup level of an account, and may not change any information within the database.

3.2. Password Requirements

When a user is creating a new account, the users selected password may be weak and easily guessable. The password will be required to include four alphabetical characters, one number, and one special character. This will help to improve the security of a user's account without altering the database. The user will be notified if the system detects that the requirements are not met.

3.3. Limit the Order

When the user logs in and requests an order for a certain number of books. The user can only order a maximum number of N (where N is set by the manager – if N is not set then there is no limit) books within a month. To limit one user from buying all the books within the bookstore, the limit can only be removed/reset by a manager. This requires that the number of books within the current order in addition to the number of books already ordered within the month is less than N. This does not edit the database when the check is performed and will not alter the database. The user will be notified by a pop-up window that the order exceeds the monthly limit.

3.4. CAPTCHA

As mentioned in requirement *2.1 Customer Registration*. When a user creates an account there is a possibility of malicious intent. A user could create a bot to create many accounts, in the attempt to delude book ratings of a certain book or to cheat the trust factor system that makes user reviews with a higher rating show first. Therefore, implementing a CAPTCHA to reduce the number of requests, consequently increasing the amount of time to register should be introduced. This can prevent bots from making and managing many accounts easily and prevent manipulation.

3.5. Random

Sometimes it is hard to find a new and interesting book, therefore when searching for a new book with no subject in mind, it should be possible for the user to randomly request a book with no additional information. This way, the user can browse the bookstore from a different point of view. This will be similar to the regular way books are searched, however only one random book from the database will be returned, nothing will be altered within the database and since it is only a query; there is no requirement to log in.

4. Conceptual Database Design

Below in *Figure 1* is the depiction of the ER-diagram for the bookstore with the requirements covered in *2. Requirement Analysis* and *3. Additional Proposed Functionality* of the report.

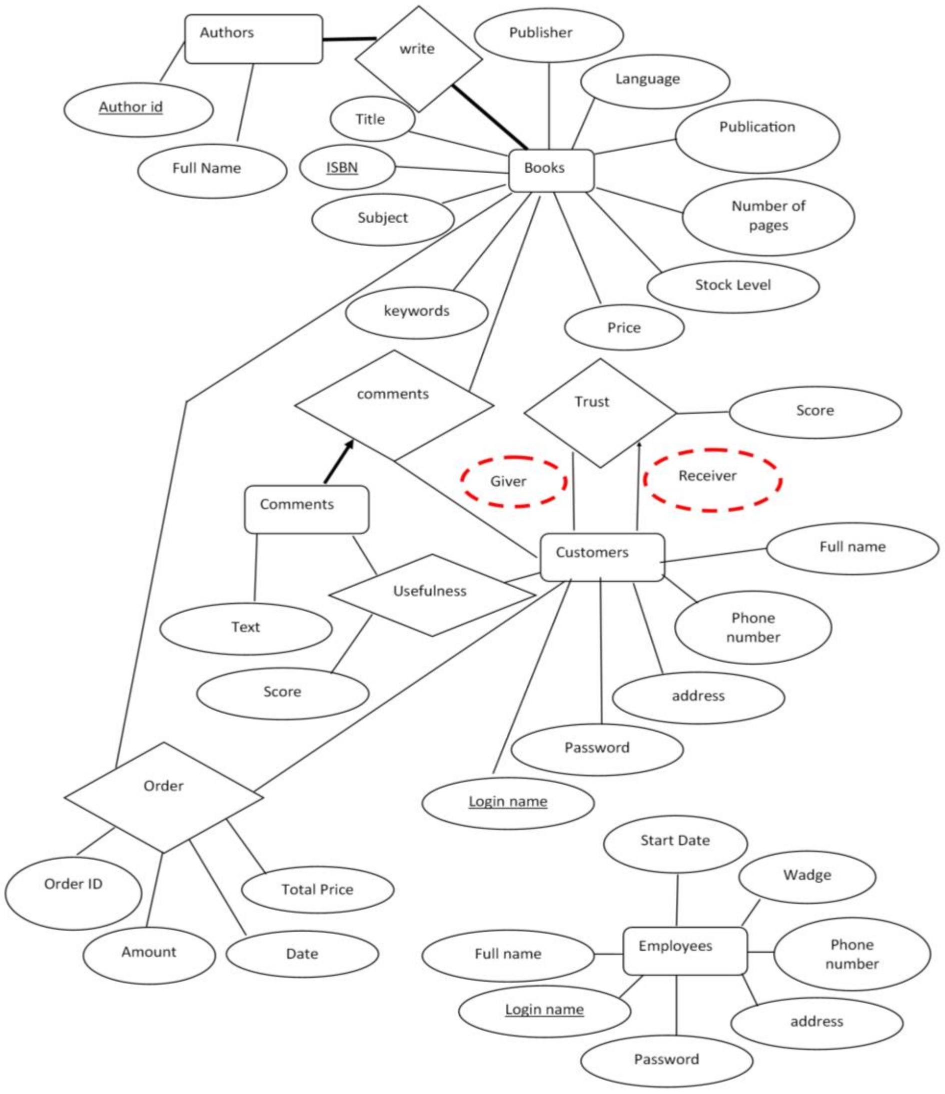

Figure 1 – The ER-Diagram for the Bookstore database

4.1. Books

The Bookstore ER Diagram for books needs to hold the following information: ISBN, title, author(s), publisher, language, publication date, number of pages, stock level (number of copies), price, keywords (used for searching genre), and subject. Within *Figure 1*, The ISBN is considered the Primary Key; since it does not change for a current book and is a unique identifier for every book.

Every book can have multiple authors. Therefore, the entity set books have a relation to the entity set authors, and the relationship is many to many, because authors can create many books, and a book can have many authors. However, every author has to be an author of at least one book and every book must have at least one author.

Figures- See Figure 1 below depicting the ER-Diagram for the Books within the database.

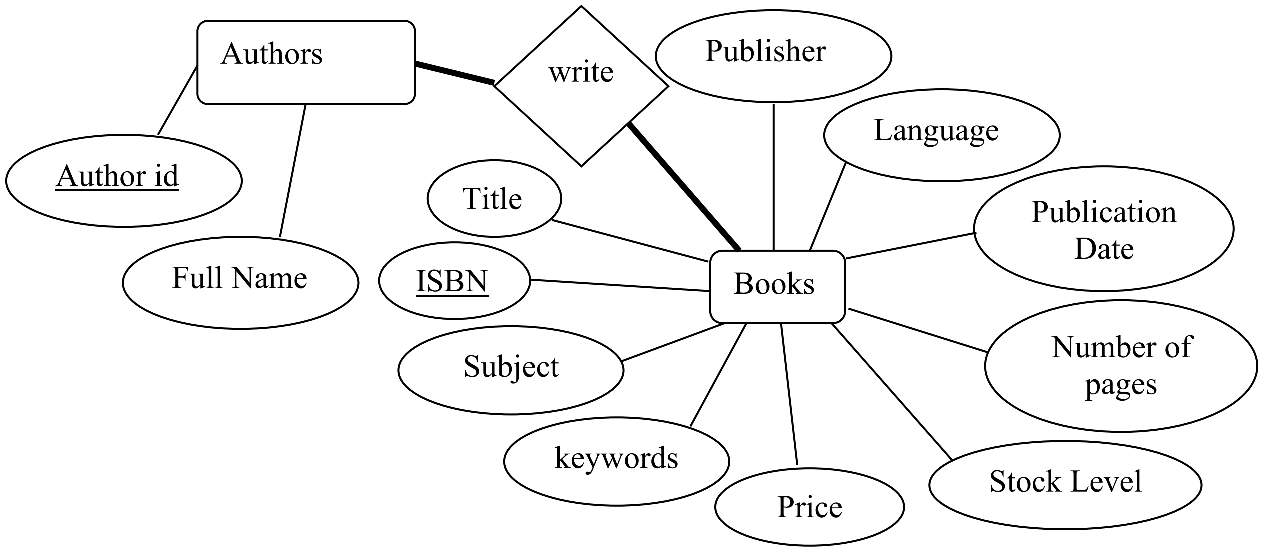

Figure 2 – The ER-Diagram for books table

Below is the data schema DDL for books; The ISBN is the Primary key and the foreign key to the author's table. The keywords are text data types where regular expressions can be used to check for the existence of certain words.

```
CREATE TABLE Books (ISBN VARCHAR(20) NOT NULL, Subject VARCHAR(255),
Keywords TEXT, Price FLOAT, StockLevel INT, NumberOfPages INT, PublicationDate
DATE, `Language` VARCHAR(255), Publisher VARCHAR(255), PRIMARY KEY (ISBN),
FOREIGN KEY (ISBN) REFERENCES Authors(ISBN);
```

Since there can be many authors for one book and an author's name may be similar to another, the primary key must be a combination of the ISBN and the AuthorID (an auto-generated number for the specific Author)

```
CREATE TABLE Authors (ISBN VARCHAR(20) NOT NULL, AuthorID INT
AUTO_INCREMENT, FullName VARCHAR(255), PRIMARY KEY (ISBN, AuthorID));
```

4.2. Customers

The Bookstore ER Diagram for Customers includes fields for full name, login name (primary key), password, address, and phone number. The login name is unique as specified in the requirements *2.1 Customer Registration* and the password is stored in a hash.

Figures- See Figure 3 below depicting the ER-Diagram for the customers within the database.

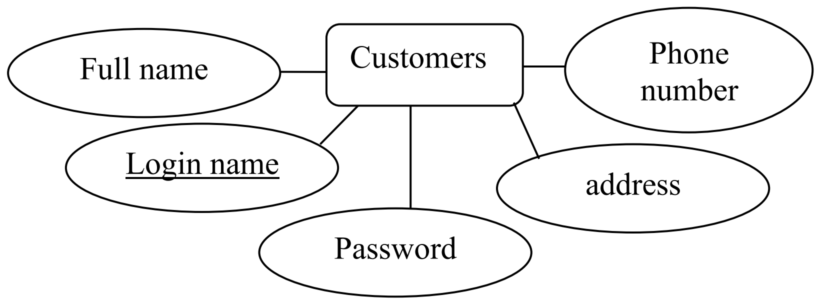

Figure 3 – The ER-Diagram for Customers table

Below is the data schema DDL for Customers. The Login name is the primary key, and every field is required to not be NULL as required in *2.1 Customer Registration* of the document.

CREATE TABLE Customers (LoginName VARCHAR(50) NOT NULL, FullName VARCHAR(255) NOT NULL, \`Password\` VARCHAR(255) NOT NULL, address TEXT NOT NULL, PhoneNumber VARCHAR(12) NOT NULL, PRIMARY KEY (LoginName));

4.3. Employees

The Bookstore ER Diagram for employees includes all relevant information gathered for customers while also holding information on wages and employee start date. The login name is unique to the employee and therefore is the Primary key and the password is stored in a hash.

The wage and start date fields will not be completed at the time of adding a new employee; however, the start date will automatically be filled in at the time the employee is added into the database and the wage can be set at any time by the manager/superuser.

Figures- See Figure 4 below depicting the ER-Diagram for the employees within the database.

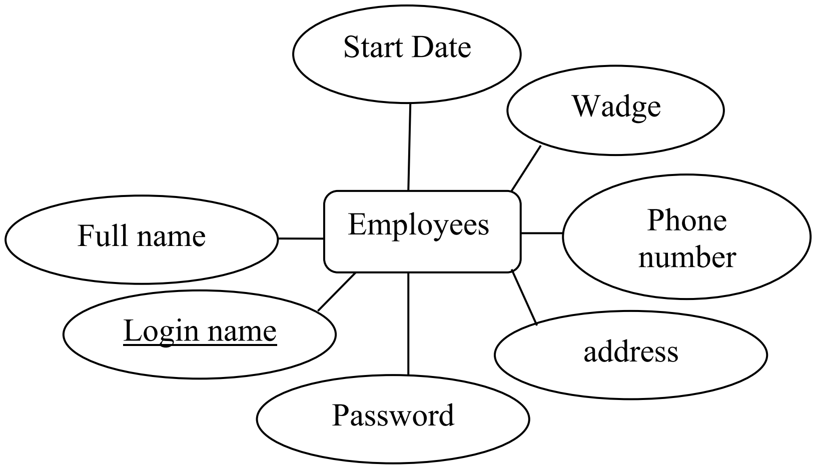

Figure 4 – The ER-Diagram for Employees

Below is the data schema DDL for Employees; like the customer's table however including wage and a start date. The employee table is separate from customers so that verification of employees logging in is quick (not querying all users) and clear.

CREATE TABLE Employee (LoginName VARCHAR(50) NOT NULL, FullName VARCHAR(255) NOT NULL, \`Password\` VARCHAR(255) NOT NULL, address TEXT NOT NULL, PhoneNumber VARCHAR(12) NOT NULL, StartDate DATE, wadge FLOAT, PRIMARY KEY (LoginName));

4.4. Comments

The Bookstore ER Diagram for Comments includes the comment is the data type TEXT (pointer to the text for the comment within the database) and the comment id, which is used as a foreign key on the usefulness table since every comment can have multiple usefulness scores given by multiple users.

Figures- See Figure 5 below depicting the ER-Diagram for the comments within the database.

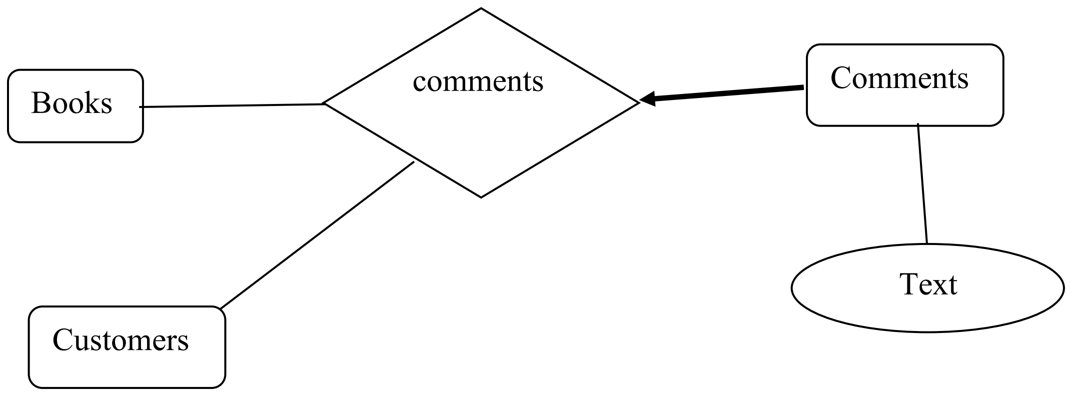

Figure 5 – The ER-Diagram for Comments

Below is the data schema DDL for comments; where the login name and ISBN are the primary keys since every user can only put one comment on every book. The comments are also deleted when either of the primary keys is removed. (a book or a user is removed)

CREATE TABLE Comments (LoginName VARCHAR(50) NOT NULL, ISBN VARCHAR(20) NOT NULL, `TEXT` TEXT, CommentID INT AUTO_INCREMENT, PRIMARY KEY (LoginName, ISBN) FOREIGN KEY (CommentID) REFERENCES Usefullness(CommentID) ON UPDATE RESTRICT ON DELETE CASCADE);

4.5. Usefulness

Below is The Bookstore ER Diagram for the usefulness score that can be given by any user to any comment. The usefulness score is given by a customer for any comment, however there may only be one score for ever comment and a user may chose to not give a score on a specific or any comment.

Figures- See Figure 5 below depicting the ER-Diagram for the Usefulness within the database.

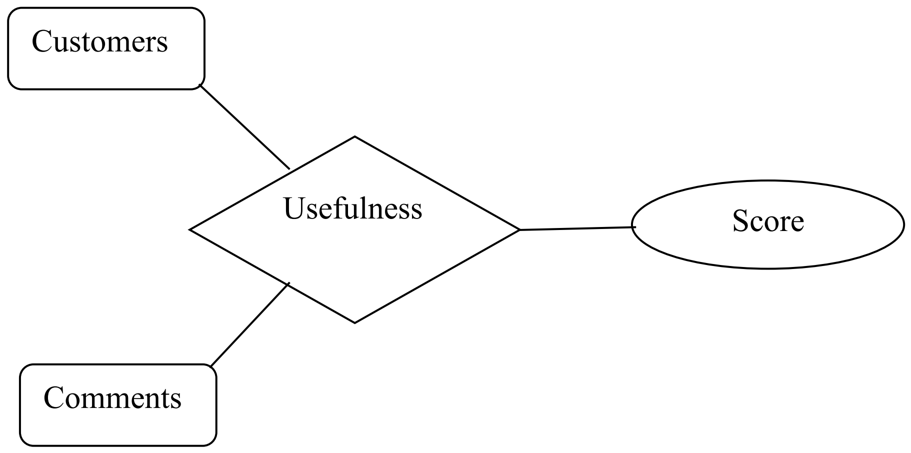

Figure 6 – The ER-Diagram for Usefulness

Below is the data schema DDL for usefulness score; since the comment score is given by any user to any comment, the primary key must be the combination of Customers and Comments.

CREATE TABLE Usefulness (LoginName VARCHAR(50) NOT NULL, CommentID INT NOT NULL, Score INT NOT NULL, PRIMARY KEY (CommentID, LoginName), ON UPDATE RESTRICT ON DELETE CASCADE);

4.6. Trust Factor

The Bookstore ER Diagram for Trust Factor, where a customer can give a trust score to another customer. The trust factor is given from one customer to another and therefore requires the relation to go back to the customer's entity set (every customer can have one score for every other customer).

Figures- See Figure 5 below depicting the ER-Diagram for the Trust Factor within the database.

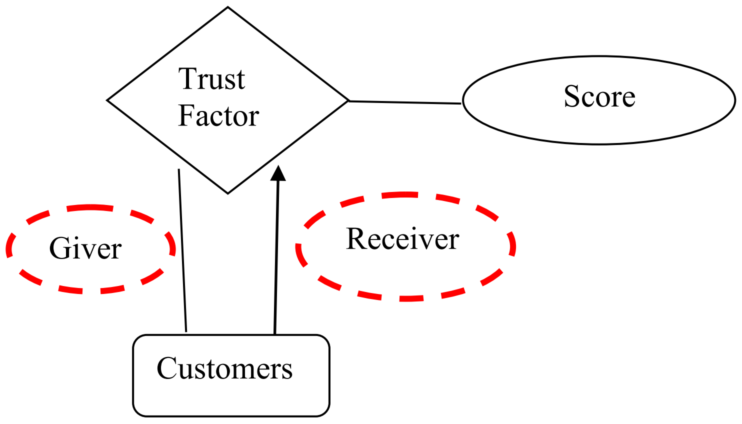

Figure 7 – The ER-Diagram for Trust Factor

Below is the data schema DDL for Trust scores; where trust scores are given from one customer to another and only one score can be given. Therefore, the primary key must be the Two Customer Login names. The trust score must also be removed if either of the users is removed from the database.

CREATE TABLE TrustScore (LoginNameGiver VARCHAR(50) NOT NULL,
LoginNameRecever VARCHAR(50) NOT NULL, Score INT NOT NULL, PRIMARY KEY
(LoginNameGiver, LoginNameRecever), ON UPDATE RESTRICT ON DELETE CASCADE);

4.6. Orders

The Bookstore ER Diagram for Orders, where a customer can choose a book's ISBN and an amount that they need and place the order for that many of that book. Orders of multiple books will be processed separately and priced accordingly to the amount for each set of books. ***Figures-*** See Figure 5 below depicting the ER-Diagram for the Orders within the database.

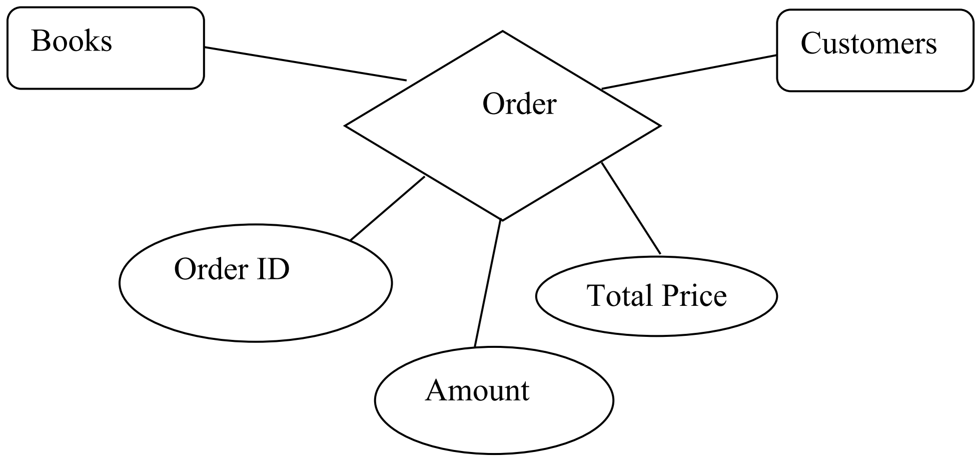

Figure 8 – The ER-Diagram for Orders

Below is the data schema DDL for Orders, where orders are between books and users. Therefore, the primary key for the Orders table is the Book's ISBN and the customer's login name. The orders table holds the Order ID so that orders that occur together can be aggregated. Then the amount for that number of books, the total price, and the data the order took place is saved.

CREATE TABLE Orders (LoginName VARCHAR(50) NOT NULL, ISBN VARCHAR(20) NOT NULL, OrderID INT AUTO_INCREMENT, `Date` DATE, amount INT NOT NULL, Price FLOAT, PRIMARY KEY (LoginName, ISBN));

5. Technology Survey

Between the three stacks LAMP, MEAN, and Django (w3schools) I have chosen for this project to implement the project with the Django stack, because of the familiarity, easy prototyping, and the extensive resources throughout the internet.

5.1. Django stack

The Django stack includes JavaScript, Python, Django, and MySQL. The Django stack includes Python, which is newer than the others. However, Python's "official documentation and tutorials are some of the best anywhere in software development." (fullstackpython) This is due to the extremely easy prototyping that python offers since the language is simple in syntax and usually taught first in schools and learning environments. Also, participating in using a stack with a constantly growing community ensures that there will always be contracting work that requires the field. Finally, some modules and frameworks are not outdated. Outdated frameworks result in unique errors that may not have community solutions.

5.2. LAMP stack

The LAMP stack includes JavaScript, Linux, Apache, MySQL, and PHP. One positive to using Linux within the stack is with Linux's virtual server (redhat), where a host database can be deployed with multiple virtual database copies. This is extremely useful for databases that are used worldwide because the locality of a virtual database may be closer to the request server and host computer resulting in a decrease in response times. However, this requires server deployment on a Linux kernel system, which is not as popular as Windows, making it more difficult to acquire a system that can support this stack. Finally, the inclusion of Apache and PHP means a higher stack than Django' and will require more time to prototype a database.

5.3. MEAN stack

The MEAN stack includes JavaScript, MongoDB, Express, AngularJS, and Node.js. The front-end development is AngularJS (a framework implemented into HTML to communicate with the server) which boasts "two-way data binding as a way to implement buttons for processing requests. This however results in a less direct approach and can be considerably slower in runtime." (mnemon1ck) The back end is taken care of by Node.js which is a fast platform "built into Chrome's JavaScript" (tutorialspoint) and is recent, developed in 2009. This supplies a very fast, asynchronous API that is 'non-blocking.' This is better than LAMP and Django for long term development and maintenance, with the additional requirement of a complete understanding of the API to take advantage of the increased speed, which could lead to increased time of prototyping.

6. Logical database design and Normalization

Every table schema (listed below) is in BCNF form because there is no bad FD. (Functional Dependency) Every Function X $\rightarrow$ Y, where X is always from the set of candidate keys and x + y includes all attributes. For table one: books, the ISBN is the Primary key, and Subject through Publisher are all related to the ISBN and not each other; this does not include trivial FD's.

The subject is not duplicate information since the subject does not determine the keywords, price, StockLevel, NumberOfPages, PublicationDate, Language, and Publisher; therefore, the subject should be within this table and this table should not be split into multiple tables. The same can be said for every other table listed below.

1) Books (ISBN, Subject, Keywords, Price, StockLevel, NumberOfPages, PublicationDate, Language, Publisher);

2) Authors (ISBN, AuthorID, FullName);

3) Customer (LoginName, FullName, Password, address, PhoneNumber);

4) Employee (LoginName, FullName, Password, address, PhoneNumber, StartDate, Wadge);

5) Comments (LoginName, ISBN, `TEXT`, CommentID);

6) TrustScore (LoginNameGiver, LoginNameRecever, Score);

7) Orders (ISBN, LoginName, OrderID, Amount, Date, TotalPrice);

7. Conclusion

This document states the requirements required by the course and includes three separate requirements proposed to be implemented. All requirements' inputs, outputs data, and side effects stated outline the functionality of the proposed system. This, along with the ER diagrams and DDL, helps to outline the structure needed to develop an implementation that meets the proposed functionality in sections two and three of the document. Finally, the technology survey and Logical database design and normalization are meant to verify that the process in PART 2 of the project is reasonable.

Source Citations

Chapter 1. Linux Virtual Server Overview Red Hat Enterprise Linux 4. access.redhat.com/documentation/en-us/red_hat_enterprise_linux/4/html/virtual_server_administration/ch-lvs-overview-vsa.

Django. www.fullstackpython.com/django.html.

mnemon1ck. *Why You Should Not Use Angularjs*. 21 Feb. 2015, medium.com/@mnemon1ck/why-you-should-not-use-angularjs-1df5ddf6fc99.

"Node.js - Introduction." *Tutorialspoint*, www.tutorialspoint.com/nodejs/nodejs_introduction.htm.

What Is Full Stack? www.w3schools.com/whatis/whatis_fullstack.asp.