Bookstore Database Design Document

Yukun Cui

Table of Contents

List of Figures

List of Tables

1. INTRODUCTION

This project implements a prototype of Bookstore Management Database System.

The main functions of this system include: User registration, Initialize managers, Order books, Add new book into stock, Add more copies of existed book, Make comments of books, Rate comments, Trust users, Browse books, Get useful comments on book, Get suggested books, Get books written by authors that are 1-degree or 2-degree separated from given author, Get book statistic, Award users, Add addresses, Add payment method, Delete comment, Delete address, Delete payment method, Cancel Order.

The database system will store information of books, customers, authors, keywords, addresses, payment methods, orders, and comments.

2. Requirement Analysis

2.1. Data

Book(

<u>ISBN</u> CHAR(13) NOT NULL,

title VARCHAR(255) NOT NULL,

publisher VARCHAR(255) NOT NULL,

language VARCHAR(20) NOT NULL,

publishDate DATETIME NOT NULL,

pageNumber INT NOT NULL,

stockLevel INT NOT NULL,

price DECIMAL(12,2) NOT NULL,

subject VARCHAR(20) NOT NULL,

PRIMARY KEY (ISBN)

FOREIGN KEY (ISBN) REFERENCES Write(ISBN)

ON DELETE CASCADE,

FOREIGN KEY (ISBN) REFERENCES Key (ISBN)

ON DELETE CASCADE,

FOREIGN KEY (ISBN) REFERENCES Put(ISBN)

ON DELETE RESTRICT,

FOREIGN KEY (ISBN) REFERENCES Comment(ISBN)

ON DELETE CASCADE,

FOREIGN KEY (ISBN) REFERENCES CommentRate(ISBN)

ON DELETE CASCADE)

Write(

ISBN CHAR(13) NOT NULL,

firstName VARCHAR(25) NOT NULL,

lastName VARCHAR(25) NOT NULL,

PRIMARY KEY (ISBN, firstName, lastName))

Author(

firstName VARCHAR(25) NOT NULL,

lastName VARCHAR(25) NOT NULL,

PRIMARY KEY (firstName, lastName),

FOREIGN KEY (firstName, lastName) REFERENCES Write(firstName, lastName))

Key(

ISBN CHAR(13) NOT NULL,

keywordType VARCHAR(25) NOT NULL,

PRIMARY KEY (ISBN, keywordType))

Keyword(

keywordType VARCHAR(25) NOT NULL,

PRIMARY KEY (keywordType),

FOREIGN KEY (keywordType) REFERENCES Key (keywordType))

Customer(

```
    loginName VARCHAR(25) NOT NULL,
    password VARCHAR(30) NOT NULL,
    firstName VARCHAR(25) NOT NULL,
    lastName VARCHAR(25) NOT NULL,
    manager BIT NOTNULL DEFAULT 0,
    PRIMARY KEY (loginName),
    FOREIGN KEY (loginName) REFERENCES Address(loginName)
    ON DELETE CASCADE,
    FOREIGN KEY (loginName) REFERENCES Payment(loginName)
    ON DELETE CASCADE,
    FOREIGN KEY (loginName) REFERENCES Put(loginName)
    ON DELETE CASCADE,
    FOREIGN KEY (loginName) REFERENCES Comment(loginName)
    ON DELETE CASCADE,
    FOREIGN KEY (loginName) REFERENCES CommentRate(loginName)
    ON DELETE CASCADE,
    FOREIGN KEY (loginName) REFERENCES CommentRate(ratingUser)
    ON DELETE CASCADE,
    FOREIGN KEY (loginName) REFERENCES Trust(loginName)
    ON DELETE CASCADE,
```

FOREIGN KEY (loginName) REFERENCES Trust(declareName)

ON DELETE CASCADE)

Note: I change address and phone number to a new table, and I also change order to another new table. If Customer.manager is 0 then the customer is not a manger. If Customer.manager is 1 then the customer is a manger.

Address(

<u>loginName</u> VARCHAR(25) NOT NULL,

<u>addID</u> INT AUTO_INCREMENT,

street VARCHAR(30) NOT NULL,

city VARCHAR(25) NOT NULL,

state VARCHAR(25) NOT NULL,

phone VARCHAR(15) NOT NULL

postalCode INT NOT NULL,

PRIMARY KEY (loginName, addID),

FOREIGN KEY (addID) REFERENCES Put(addID)

ON UPDATE RESTRICT ON DELETE RESSTRICT)

Note: phone number could be +1(123)456-7890 or (123)456-7890 or 1234567890.

Payment(

<u>loginName</u> VARCHAR(25) NOT NULL,

<u>cardNumber</u> INT NOT NULL,

expireDate DATETIME NOT NULL,

PRIMARY KEY (loginName, cardNumber),

FOREIGN KEY (loginName, cardNickName) REFERENCES Put(cardNumber)

ON UPDATE RESTRICT ON DELETE RESSTRICT)

Put(

loginName VARCHAR(25) NOT NULL,

ISBN CHAR(13) NOT NULL,

orderNumber VARCHAR(25) NOT NULL,

addID INT NOT NULL,

cardNumber INT NOT NULL,

PRIMARY KEY (loginName, ISBN, orderNumber))

Order(

orderNumber VARCHAR(25) NOT NULL AUTO INCREMENT,

amount INT NOT NULL,

PRIMARY KEY (orderNumber),

FOREIGN KEY (orderNumber) REFERENCES Put(orderNumber)

ON UPDATE RESTRICT ON DELETE CASCADE)

Comment(

loginName VARCHAR(25) NOT NULL,

ISBN CHAR(13) NOT NULL,

content VARCHAR(255) DEFAULT NULL

score INT NOT NULL,

PRIMARY KEY (loginName, ISBN),

FOREIGN KEY (loginName, ISBN) REFERENCES Comment(loginName, ISBN)

ON DELETE CASCADE)

CommentRate(

loginName VARCHAR(25) NOT NULL,

ISBN CHAR(13) NOT NULL,

ratingUser VARCHAR(25) NOT NULL

rating VARCHAR(11) NOT NULL,

PRIMARY KEY (loginName, ISBN, ratingUser))

Trust(

loginName VARCHAR(25) NOT NULL,

declareName VARCHAR(25) NOT NULL

trustORnot VARCHAR(11) NOT NULL,

PRIMARY KEY (loginName, declareName))

Note: trustORnot can be “trusted”, “not-trusted” or NULL.

2.2. Functions

1. **CustomerRegistration**

Client: User.

Input: loginName, password, firstName, lastName

Output: if loginName is not taken, return "Success"; otherwise return "Login Name is already taken."

Side effects: If output is "Success", then a new record will be added to table Customer.

Constrains: loginName must be unique, password, firstName, lastName must meet the same constrains described in table Customer.

2. **ManagerRegistration**

Client: Manager.

Input: loginName, password, firstName, lastName, manager

Output: if loginName is not taken, return "Success"; otherwise return "Login Name is already taken."

Side effects: If output is "Success", then a new record will be added to table Customer. And Customer.manager of this record will be set to 1.

Constrains: loginName must be unique, password, firstName, lastName must meet the same

constrains described in table Customer. The first manger is the first customer starts this system.

3. **Ordering**

Client: User.

Input: loginName, ISBN, amount, addID, cardNickName

Output: If records in Customer, Book, Address and Payment exists: If the stock level of the ISBN is not 0 and not smaller than amount, then return "Success". If the stock level of the ISBN is not 0 and smaller than amount, then return "Not enough in stock". If the stock level of the ISBN is 0, then return “Sold Out”. Otherwise: If the loginName does not exist, then return “Invalid login”. If the ISBN does not exist, then return “No such book”. If the addID does not exist, then return “Invalid address”. If the cardNickName does not exist, then return “Invalid payment method”.

Side effects: If the output is “Success”, then the stockLevel of this book will decrease by amount, create, and a new record will be created in table Order.

Constrains: loginName, ISBN, addID, cardNickName must meet the table constrains.

4. **NewBook**

Client: Manager.

Input: ISBN, title, publisher, language, publishDate, pageNumber, amount, author[], price, keyword[], subject

Output: If the input ISBN does not exist, then return “Success”. If the input ISBN exists, then return “Book already exists”.

Side effects: If the output is “Success”, then the create new record in Book with given information except author and keyword. For authors and keywords, input are 2 arrays. For each element, if already exists in Author and Keyword, then just create new record in Write and Key with the same ISBN. Otherwise create new record in Author and Keyword and then create new record in Write and Key with the same ISBN.

Constrains: ISBN, title, publisher, language, publishDate, pageNumber, amount, author[], price,

keyword[], subject must meet the table constrains. ISBN must be unique.

5. **NewCopy**

Client: Manager.

Input: ISBN, amount.

Output: If the input ISBN exists, then return “Success”. If the input ISBN does not exist, then return “Book does not exist”.

Side effects: If the output is “Success”, then stockLevel of the record in Book with given ISBN will increase by amount.

Constrains: ISBN, amount(stockLevel) must meet the table constrains.

6. **AddComment**

Client: User.

Input: ISBN, loginName, content, score.

Output: If the input ISBN and loginName exist, then return “Success”. If the input ISBN does not exist, then return “Book does not exist”. If the input loginName does not exist, then return “User does not exist”.

Side effects: If the output is “Success”, then the create new record in Comment with given information.

Constrains: ISBN, loginName, content, score must meet the table constrains. Score must be an integer from 0 to 10. (loginName, ISBN) must be unique.

7. **CommentRating**

Client: User.

Input: loginName, ISBN, ratingUser, rating.

Output: If record in table Comment with primary key equal to (loginName, ISBN), then return “Success”. If no match found, then return “No such comment”.

Side effects: If the output is “Success”, then the create new record in table CommentRate with (loginName, ISBN, ratingUser, rating) as (loginName, ISBN, ratingUser, rating).

Constrains: loginName, ISBN, ratingUser, rating must meet the table constrains. Rating must be “useless”, “useful” or “very useful”. (loginName, ISBN, ratingUser) must be unique.

8. **TrustRecording**

Client: User.

Input: loginName, declareName, trustORnot

Output: If there exists record in Customer matches loginName as loginName and there also exists record in Customer matches declareName as loginName, then return “Success”. If there does not exist record in Customer matches loginName as loginName, then return “The customer you want to set trust record on does not exist”. If there does not exist record in Customer matches declareName as loginName, then return “You cannot set trust record without registration”.

Side effects: If the output is “Success”, then the create new record in table Trust with given information.

Constrains: loginName, declareName, trustORnot must meet the table constrains. trustORnot must be “trusted” or “not-trusted”. (loginName, declareName) must be unique.

9. **BookBrowsing**

Client: User.

Input: ifauthor, author, ifpublisher, publisher, iftitle, title, iflanguage, language, ifdate, ifscore, iftrust.

Output: If ifauthor is True, then return records in Book that matches author. If ifpublisher is True, then return records in Book that matches publisher. If iftitle is True, then return records in Book that matches title. If iflanguage is True, then return records in Book that matches language. This can be implemented using subqueries. If ifdate is true, then sort the result with publishDate. If ifdate is true, then sort the result with average comment score. If iftrust is true, then sort the result with average comment score from trusted customers. This can be implemented with ODER BY, GROUP BY, JOIN, AVERAGE, and subqueries.

Side effects: This function only show results, it will not change the database.

Constrains: ifauthor, ifpublisher, iftitle, iflanguage, ifdate, ifscore, iftrust must be not-NULL Boolean values. author, publisher, title, language must meet the table constrains if their related Boolean value is True.

10. **UsefulComment**

Client: User.

Input: ISBN, n

Output: If the input ISBN exists, then return the top n useful comments. This can be implemented by JOIN Book and Comment and CommentRate. Use GROUP BY , AVERAGE and ODER BY to return the result. If the input ISBN does not exist, then return “Book already exists”.

Side effects: This function will not change the database.

Constrains: ISBN must meet the table constrains. n must be an integer.

11. **BuyingSuggestion**

Client: User.

Input: ISBN

Output: If the input ISBN exists, then JOIN Customer and Book and Order. Find all records of books that purchased by same users who purchased book with input ISBN. And use COUNT to count their sales amount.

Then return all attributes in table Book of records that satisfies such requirements, ordered by result of count in descending order. If the input ISBN does not exist, then return “No such book”.

Side effects: This function will not change the database.

Constrains: ISBN must meet the table constrains.

12. **Degree**

Client: User.

Input: author, degree

Output: If the input author exists: If degree is 1, then return all attributes that written by authors who are 1-degree separated with input author. If degree is 2, then return all attributes that written by authors who are 2-degree separated with input author. Otherwise, if the input author does not exist, then return “No such author”.

Side effects: This function will not change the database.

Constrains: author must meet the table constrains. degree must be an integer and its value must be

1 or 2.

13. **BookStatistics**

Client: Manager.

Input: m.

Output: Return top m records of Book ordered by sum of amount in orders of this book in descending order. Return m records of author ordered by sum of amount in orders of books written by this author in descending order. Return m records of publisher ordered by sum of amount in orders of books published by this publisher in descending order.

Side effects: This function will not change the database.

Constrains: m must be an integer.

14. **UserAwards**

Client: Manager.

Input: m

Output: Return top m records of customer ordered by count number in Trust that this customer is set to be "trusted" in descending order. Return top m records of customer ordered by average rating of comment made by this customer in descending order.

Side effects: This function will not change the database.

Constrains: m must be an integer.

15. **NewAddress**

Client: User.

Input: loginName, street, city, state, phone, postalCode

Output: If the input loginName exists, then return "Success". If the input loginName does not exist, then return "Invalid loginName".

Side effects: If the output is "Success", then the create new record in Address with given information. And addID will automatically increase.

Constrains: loginName, street, city, state, phone postalCode must meet the table constrains. phone number could be like +1(123)456-7890 or (123)456-7890 or 1234567890. (loginName, addID) must be unique.

16. **PaymentMethod**

Client: User.

Input: loginName, cardNumber, expireDate

Output: If the input loginName exists and no record in Payment with this loginName have the same cardNumber as input cardNumber exists, then return “Success”. If the input loginName does not exist, then return “Invalid loginName”. If there is record in Payment with this loginName have the same cardNumber as input cardNumber, then return “Card has been registered”.

Side effects: If the output is “Success”, then the create new record in Payment with given information.

Constrains: loginName, cardNumber, expireDate must meet the table constrains. (loginName, cardNumber) must be unique.

17. **deleteComment**

Client: User.

Input: loginName, ISBN.

Output: If records in Comment has the primary key matches input (loginName, ISBN), return “Success”. Otherwise return “Comment does not exist”.

Side effects: If the output is “Success”, then the delete the related record in Comment, as well as records in CommentRate that point to these deleted records.

Constrains: loginName, ISBN must meet the table constrains

18. **DeleteAddress**

Client: User.

Input: loginName, addID.

Output: If records in Address has the primary key matches input (loginName, addID) and if there is no record in Order that point to these records, return “Success”. If there is no record in Address has the primary key matches input (loginName, addID), return “Address does not exist”. If there are record in Order that point to these records, return “Address used in existed orders cannot be deleted”.

Side effects: If the output is “Success”, then the delete the related record in Address.

Constrains: loginName, addID must meet the table constrains.

19. **DeletePaymentMethod**

Client: User.

Input: loginName, cardNumber.

Output: If records in Payment has the primary key matches input (loginName, cardNumber) and if there is no record in Order that point to these records, return “Success”. If there is no record in Payment has the primary key matches input (loginName, cardNumber), return “Payment method does not exist”. If there are record in Order that point to these records, return “Payment method used in existed orders cannot be deleted”.

Side effects: If the output is “Success”, then the delete the related record in Payment.

Constrains: loginName, cardNumber must meet the table constrains.

20. **CancelOrder**

Client: User.

Input: orderNumber.

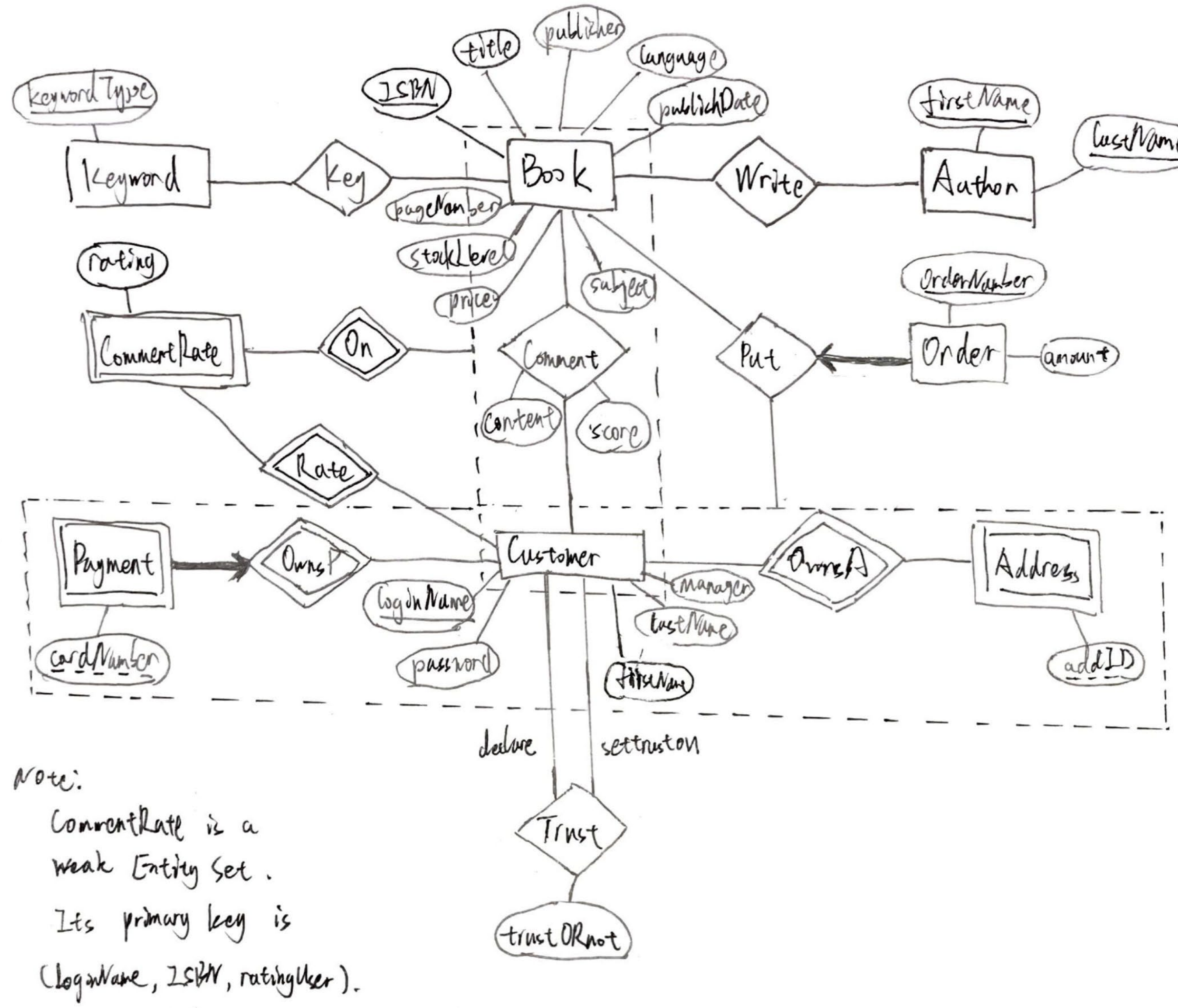
Output: If records in Order matches input orderNumber exists, return “Success”. Otherwise, return “Oder does not exist”.

Side effects: If the output is “Success”, then the delete the related record in Order. Then also delete the related record with same orderNumber in Put since the foreign key in Order is set “ON DELETE CASCADE”.

Constrains: orderNumber must meet the table constrains.

3. Conceptual Database Design

Figures- Figure 1 below is the ER diagram describing my conceptual database design



Note:
CommentRate is a
weak Entity Set.
Its primary key is
(loginName, ISBN, ratingUser).
loginName, ISBN comes from Relationship Set: Book. Comment, Customer,
ratingUser comes from Customer.

Figure 1 - ER Diagram

4. Technology Survey

4.1. Python Flask, SQLite

Python Flask: Flask is a micro web framework written in Python[1]. Python flask is popular for web application. Flask is flexible, light-weight and it supports many extensions such as Flask-SQLAlchemy. Flask-SQLAlchemy enables us to develop software that can manage SQL database, which fits the purpose of this project. Python is easy and light to use. Since our project is an individual project and also a prototype, python flask is the most efficient way to implement this prototype application.

SQLite is a relational database management system contained in a C library[2]. SQLite is simpler and MySQL is more advanced. Since we have learned SQL queries and advanced SQL queries, it is more efficient to use SQLite to develop prototype application in limited time.

4.2. JavaScript, MySQL

JavaScript is a high-level, often just-in-time compiled, and multi-paradigm programming language[3]. It is one of the most popular programming languages used in web application development due to its cast library of classes. However, application using JavaScript usually involves multiple developers. The work will be too heavy for individual project and not necessary for a prototype application.

MySQL is an open-source relational database management system[4]. MySQL is more advanced than SQLite but as I said in 4.1, SQLite is enough for this project.

4.3. Python Django, SQLite

Django is a Python-based free and open-source web framework that follows the model-template-views architectural pattern[5]. Django is a full-stack web frame, so it is much “heavier” than flask. After requirement analysis I think flask is enough for this project.

5. Logical database design and Normalization

Book

Table 1 - Book

<u>ISBN</u>	title	publisher	language	publishDate	pageNumber	stockLevel	price	subject

Table 1 - Book

ISBN → ISBN, title, publisher, language, publishDate, pageNumber, stockLevel, price, subject

(ISBN) is a key of Book. No bad FD.

Write

Table 2 - Write

<u>ISBN</u>	<u>firstName</u>	<u>lastName</u>

Table 2 - Write

ISBN, firstName, lastName → ISBN, firstName, lastName

(ISBN, firstName, lastName) is a key of Write. No bad FD.

Author

Table 3 - Author

firstName	lastName

Table 3 - Author

firstName, lastName → firstName, lastName

(firstName, lastName) is a key of Author. No bad FD.

Key

Table 4 - Key

ISBN	keywordType

Table 4 - Key

ISBN, keywordType → ISBN, keywordType

(ISBN, keywordType) is a key of Key. No bad FD.

Keyword

Table 5 - Keyword

keywordType

Table 5 - Keyword

keywordType → keywordType

(keywordType) is a key of Keyword. No bad FD.

Customer

Table 6 - Customer

loginName	password	firstName	lastName	manager

Table 6 - Customer

loginName → loginName, password, firstName, lastName, manager

(loginName) is a key of Customer. No bad FD.

Address

Table 7 - Address

loginName	addID	street	city	state	phone	postalCode

Table 7 - Address

loginName, addID → loginName, addID, street, city, state, phone, postalCode

(loginName, addID) is a key of Address. No bad FD.

Payment

Table 8 - Payment

loginName	cardNumber	expireDate

Table 8 - Payment

loginName, cardNumber → loginName, cardNumber, expireDate

(loginName, cardNumber) is a key of Payment. No bad FD.

Put

Table 9 - Put

loginName	ISBN	orderNumber	addID	cardNumber

Table 9 - Put

loginName, ISBN, orderNumber → loginName, ISBN, orderNumber, addID, cardNumber

(loginName, ISBN, orderNumber) is a key of Put. No bad FD.

Order

Table 10 - Order

orderNumber	amount

Table 10 - Order

orderNumber → orderNumber, amount

(orderNumber) is a key of Order. No bad FD.

Comment

Table 11 - Comment

loginName	ISBN	content	score

Table 11 - Comment

loginName, ISBN → loginName, ISBN, content, score

(loginName, ISBN) is a key of Comment. No bad FD.

CommentRate

Table 12 - CommnetRate

loginName	ISBN	ratingUser	rating

Table 12 - CommentRate

loginName, ISBN, ratingUser → loginName, ISBN, ratingUser, rating

(loginName, ISBN, ratingUserd) is a key of CommentRate. No bad FD.

Trust

Table 13 - Trust

loginName	declareName	trustORnot

Table 13 - Trust

loginName, declareName → loginName, declareName, trustORnot

(loginName, declareName) is a key of Trust. No bad FD.

Thus, there is no bad FD in all relations, the final schemas is in the Boyce Codd Normal Form.

6. Schedule

Week 1: Database Design

Week 2: Backend Design

Week 3: Testing

Week 4: Frontend Design

Acknowledgement

[1] Wikipedia, https://en.wikipedia.org/wiki/Flask_(web_framework)

[2] Wikipedia, https://en.wikipedia.org/wiki/SQLite

[3] Wikipedia, https://en.wikipedia.org/wiki/JavaScript

[4] Wikipedia, https://en.wikipedia.org/wiki/MySQL

[5] Wikipedia, https://en.wikipedia.org/wiki/Django_(web_framework)