Final Project Report

CSE 3241, SU 2017

Group: Haoyan Xiang, Jeanie Chen, Yuanzhe Li, Zhe Luan

SECTION 1 - Database Description

1. The final version of ER-model and relational schema please refer to the appendix 1 and 2.

2. The description of the functional dependency in each table.

(1) CUSTOMER_HAS_ADDRESS:
email -> zipcode
zipcode -> {street}
For the table CUSTOMER_HAS_ADDRESS, email determines street and
determines street.

(2) ADDRESS:
zipcode -> {street}
For the table ADDRESS, zipcode determines street.

(3) CUSTOMER_HAS_PAYMENT:
{email, credit_card} -> {email, credit_card}
For the table CUSTOMER_HAS_PAYMENT, email and credit_card determine themselves.

(4) CUSTOMER:
email -> {phone_num, Fname, Lname}
For the table CUSTOMER, email determines phone_num, Fname and Lname.

(5) PAYMENT:
credit_card -> {expired_date, security_code, zipcode}
For the table PAYMENT, credit_card determines expired_date, security_code and zipcode.

(6) PUBLISHER:
name -> phone_num
For the table PUBLISHER, Name determines phone_num.

(7) REVIEW:
r_id -> {rating, r_content, Cmail, ISBN}
For the table REVIEW, r_id determines rating, r_content, Cmail and ISBN.

(8) INVENTORY_STORE_BOOK:
{ISBN, in_id} -> amount

For the table INVENTORY_STORE_BOOK, ISBN and in_id determine amount.

(9) INVENTORY:
inventory_id -> {zipcode, amount, street}
For the table INVENTORY, inventory_id determines zipcode, amount and street.

(10) AUTHOR:
phone_num -> {Fname,Lname,Mname}
For the table AUTHOR, phone_num determines Fname,Lname and Mname.

(11) AUTHOR_WRITE_BOOK:
{ISBN, Aphone_num} -> {ISBN, Aphone_num}
For the table AUTHOR_WRITE_BOOK, ISBN and Aphone_num determine themselves.

(12) BOOK:
ISBN -> {Pname, title, price, edition, publish_date, sales_amount}
For the table BOOK, ISBN determines Pname, title, price, edition, publish_date and sales_amount.

(13) TRANSACTION_HISTORY:
transaction_id -> {Cmail, date, sold, return_date, returned, amount}
For the table TRANSACTION, transaction_id determines Cmail, date, sold, return_date, returned and amount.

(14) BOOK_HAS_TRANSACTION:
{ISBN,transaction_id} -> {ISBN, transaction_id}
For the table BOOK_HAS_TRANSACTION, ISBN and transaction_id determine themselves.

(15) POST_COURSE:
{prerequisite_num, post_course_num} -> {prerequisite_num, post_course_num}
For table POST_COURSE, prerequisite_num and post_course_num determine prerequisite_num and post_course_num

(16) COURSE_USE_BOOK:
{course_num,ISBN} -> {ISBN, course_num}
For the table COURSE_USE_BOOK, course_num and ISBN determine themselves.

(17) BOOK_BELONGS_TO_CATEGORY:
{ISBN,category_name} -> {ISBN,category_name}
For the table BOOK_BELONGS_TO_CATEGORY, ISBN and catergory_name determine ISBN and category_name.

(18) CATEGORY:
name

There's no functional dependency in the CATEGORY table.

(19) COURSE:
course_num
There is no functional dependency in the COURSE table.

4. Description of the level of normalization achieved for each table.

(1) CUSTOMER_HAS_ADDRESS: BCNF
Zipcode determines street. Email determines zipcode. Since zipcode and email are both superkeys, this table should be in BCNF.

(2) ADDRESS: BCNF
Zipcode determines street. Since zipcode is a superkey, this table should be in BCNF.

(3) CUSTOMER_HAS_PAYMENT: BCNF
Email and credit_card determine email and credit_card_num. Since {email, credit_card_num} is a candidate key, thus a superkey, this table should be in BCNF.

(4) CUSTOMER: BCNF
Email determines phone_num, Fname and Lname. Since email is a superkey, this table should be in BCNF.

(5) PAYMENT: BCNF
Credit_card determines expired_date, security_code, zipcode, and street. Since credit_card is a superkey, this table should be in BCNF.

(6) PUBLISHER: BCNF
Name determines Phone_num. Since Name is a superkey, this table should be in BCNF.

(7) REVIEW: BCNF
R_id determines rating, r_content, Cmail, and ISBN. Since r_id is a superkey, this table should be in BCNF.

(8) INVENTORY_store_BOOK: BCNF
ISBN and in_id determine amount. Since {ISBN, in_id } is a candidate key, thus a superkey, this table should be in BCNF.

(9) INVENTORY: BCNF
Inventory_id determines zipcode, amount, and street. Since inventory_id is a superkey, this table should be in BCNF.

(10) AUTHOR: BCNF
Phone_num determines Fname, Lname, and Mname. Since phone_num is a superkey, this table should be in BCNF.

(11) AUTHOR_write_BOOK: BCNF
ISBN and Aphone_num determine ISBN and Aphone_num. Since {ISBN, Aphone_num} is a candidate key, thus a superkey, this table should be in BCNF.

(12) BOOK: BCNF
ISBN determines Pname, title, price, edition, publish_date and sales_amount. Since ISBN is a superkey, this table should be in BCNF.

(13) TRANSACTION_HISTORY: BCNF
Transaction_id determines Cmail, date, sold, return_date, returned and amount. Since transaction_id is a superkey, this table should be in BCNF.

(14) BOOK_has_TRANSACTION: BCNF
ISBN and transaction_id determine themselves. Since {ISBN, transaction_id} is a candidate key, thus a superkey, this table should be in BCNF.

(15) POST_COURSE: 3NF
Prerequisite_num and post_course_num determine themselves. Since {prerequisite_num, post_course_num} is a candidate key, thus a superkey, this table should be in BCNF.

(16) COURSE_use_BOOK: BCNF
Course_num and ISBN determine course_num and ISBN. Since {ISBN, course_num} is a candidate key, thus a superkey, this table should be in BCNF.

(17) BOOKS_belongs_to_CATEGORY: BCNF
ISBN and category_name determine ISBN and category_name. Since {ISBN, category_name} is a candidate key, thus a superkey, this table should be in BCNF.

(18) CATEGORY:
Table CATEGORY has only one attribute name. Thus, no functional dependency exists in this table.

(19) COURSE:
Table COURSE has only one attribute Course_num. Thus, no functional dependency exists in this table.

5. Description of each of the indexes that implement on the database.

For the table CUSTOMER_has_ADDRESS, Hash-based Indexes will be used since equality test against zipcode, street and email in this table will be performed.

For the table ADDRESS, Hash-based Indexes will be used since equality test against zipcode and street will be performed.

For the table CUSTOMER_has_PAYMENT, Hash-based Indexes will be used since equality test against email and credit_card will be performed.

For the table CUSTOMER, Hash-based Indexes will be used since equality test against email, phone_num, Lname and Fname will be performed.

For the table PAYMENT, Hash-based Indexes will be used since equality test against credit_card, expired_date, security_code and zipcode will be performed.

For the table PUBLISHER, Hash-based Indexes will be used since equality test against name and phone_num will be performed.

For the table REVIEW, Tree-based Indexes will be used since ranged tests based on books' ratings will be performed. i.e. find books that have at least 3-star ranking.

For the table INVENTORY_store_BOOK, Tree-based Indexes will be used since ranged test based on amount will be performed. i.e. finding books that have less than 100 copies in stock.

For the table INVENTORY, Hash-based Indexes will be used since ranged tests based on amount will be performed. i.e. find books which have less than 300 copies in inventory.

For the table AUTHOR, Hash-based Indexes will be used since equality test against phone_num, Fname, Mname and Lname will be performed.

For the table AUTHOR_write_BOOK, Hash-based Indexes will be used since equality test against ISBN and Aphone_num will be performed.

For the table BOOK, Hash-based Indexes will be used since equality test against ISBN will be performed.

For the table TRANSACTION, Hash-based Indexes will be used since equality test based on transaction_id will be performed.

For the table BOOK_has_TRANSACTION, Hash-based Indexes will be used since equality test based on ISBN and transaction_id will be performed.

For the table POST_COURSE, Hash-based Indexes will be used since equality test based on prerequisite_num and post_course_num will be performed.

For the table COURSE_use_BOOK, Hash-based Indexes will be used since equality test against course_num and ISBN will be performed.

For the table BOOKS_belongs_to_CATEGORY, Hash-based Indexes will be used since equality test against ISBN and category_name will be performed.

For the table CATEGORY, Hash-based Indexes will be used since equality test based on name will be performed.

6. The BookStore database management system provide following views. These views are presented in the form of SQL statements with their relational algebraic expressions and their sample output.

i. A brief description in English of what this view produces, and why it would be useful.
ii. Relational algebra expression to produce this view.
iii. SQL statements to produce the view.
iv. Sample output from the view, with 5-10 lines of data records shown.

(a) Provide a list of customer names, along with the total dollar amount each customer has spent.

```
CREATE VIEW money_spent(Fist_name, Last_name, dollar_spent)
AS SELECT C.Fname, C.Lname, sum(T.amount * B.price)
FROM CUSTOMER C, BOOK B, TRANSACTION_HISTORY T, BOOK_HAS_TRANSACTION BT
WHERE C.email = T.Cmail
        AND T.transaction_id = BT.transaction_id AND B.ISBN = BT.ISBN
GROUP BY C.email
```

Sample output:

Fist_name	Last_name	dollar_spent
Elizabeth	White	2348
Brutus	Buckeye	2660.839999999999
James	Smith	25
Andrew	Moore	251.54999999999998
Laura	Lee	19.95
Jennifer	Harris	600
Michael	Jones	104
Susan	Thompson	442
Maria	Martin	639.2
Paul	Taylor	260
Barbara	Jackson	584.35
Charles	Wilson	38
Jones	Johnson	51.9
Robert	Williams	39.599999999999994
Linda	Thomas	504
David	Davis	83.69999999999999
Richard	Miller	168.93
William	Brown	64.95
Mary	Anderson	165
Sharon	Clark	7.99
Karen	Garcia	25

(b) Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.

```
CREATE VIEW more_than_average(First_name, Last_name, Email_Address)
        AS SELECT Fname, Lname, email
FROM (SELECT Fname, Lname, email, sum(T.amount * B.price) as cost
FROM CUSTOMER C, BOOK B, TRANSACTION_HISTORY T, BOOK_HAS_TRANSACTION BT
WHERE C.email = T.Cmail AND T.transaction_id = BT.transaction_id AND B.ISBN = BT.ISBN
GROUP BY C.email),
(SELECT avg(dollar_spent) AS avg_spent
        FROM money_spent)
WHERE  cost > avg_spent;
```

Sample output

First_name	Last_name	Email_Address
Elizabeth	White	bplwmkxwpwpnbyi@inbox.com
Brutus	Buckeye	brutusbuckeye@gmail.com
Jennifer	Harris	jvtsduaaldqzvkd@outlook.com
Susan	Thompson	lakesbnzybvgxrd@hotmail.com
Maria	Martin	ldpcnyrfhqcbljg@inbox.com
Barbara	Jackson	lsgesltuxjqsbyz@yahoo.com
Linda	Thomas	qujjcalmqpqkhwm@mail.com

(c) Provide a list of the titles in the database and associated total copies sold to customers, sorted from the title that has sold the most individual copies to the title that has sold the least.

```
CREATE VIEW BEST_SELL
AS SELECT B.title, SUM(T.amount) AS sales
FROM BOOK AS B, TRANSACTION_HISTORY AS T,BOOK_has_TRANSACTION AS BHT
WHERE T.sold = 'TRUE'
        AND T.transaction_id = BHT.transaction_id
        AND BHT.ISBN = B.ISBN
GROUP BY BHT.ISBN
ORDER BY sales DESC;
```

Sample output

title	sales
The Vanished Man: A Lincoln Rhyme Novel	35
Calculus for Scientists and Engineers: Early Transcendentals	32
The Second Time Around	27
The Guardian	16
Building Construction Illustrated	15
ON FOOD AND COOKING	14
A New Kind of Science	13
The Future of Life	12
Consilience: The Unity of Knowledge	11
House of Leaves	11
The Rescuer: The O'Malley Series	10
Linked: The New Science of Networks	10
The Blank Slate: The Modern Denial of Human Nature	9
Letters to Kelly	8
The Dark Highlander	7
Bel Canto: A Novel	6
The Five Dysfunctions of a Team: A Leadership Fable	6
Twelve Times Blessed	4
Going Postal	1
The Color of Magic	1
Guards! Guards!	1
Small Gods	1
Unseen Academicals	1
Cosmopolis: A Novel	1
Dreamcatcher	1

(d) Provide a list of the titles in the database and associated dollar totals for copies sold to customers, sorted from the title that has sold the highest dollar amount to the title that has sold the smallest.

```
CREATE VIEW BEST_PROFITS
AS SELECT B.title, sum(T.amount * B.price) AS profits
FROM BOOK AS B, TRANSACTION_HISTORY AS T, BOOK_has_TRANSACTION AS BT
WHERE T.sold = 'TRUE'
        AND T.transaction_id = BT. transaction_id
```

```
        AND BT.ISBN = B.ISBN
GROUP BY BT.ISBN
ORDER BY profits DESC;
```

Sample output

title	profits
Calculus for Scientists and Engineers: Early Transcendentals	3840
The Vanished Man: A Lincoln Rhyme Novel	875
The Second Time Around	702
Building Construction Illustrated	600
A New Kind of Science	584.35
The Guardian	399.2
ON FOOD AND COOKING	308
The Future of Life	264
Linked: The New Science of Networks	260
The Blank Slate: The Modern Denial of Human Nature	251.54999999999998
House of Leaves	219.45
Consilience: The Unity of Knowledge	165
The Rescuer: The O'Malley Series	129.9
Twelve Times Blessed	103.8
Bel Canto: A Novel	83.69999999999999
The Five Dysfunctions of a Team: A Leadership Fable	79.19999999999999
The Dark Highlander	48.93
Letters to Kelly	38
Cosmopolis: A Novel	25
The Color of Magic	13.99
Going Postal	7.99
Guards! Guards!	7.99
Small Gods	7.99
Unseen Academicals	7.99
Dreamcatcher	7.99

(e) Find the most popular author in the database (i.e. the one who has sold the most books)

```
CREATE VIEW MOST_POPULAR_AUTHOR
AS SELECT A_1.Fname, A_1.Mname, A_1.Lname, A_1.phone_num
FROM (SELECT max(B.sales_amount) AS max_sales
FROM BOOK AS B, AUTHOR AS A, AUTHOR_WRITE_BOOK AS AB
WHERE B.ISBN = AB.ISBN
AND AB.Aphone_num = A.phone_num) AS R, BOOK AS B_1, AUTHOR AS A_1,
AUTHOR_WRITE_BOOK AS AB_1
WHERE B_1.ISBN = AB_1.ISBN
    AND AB_1.Aphone_num = A_1.phone_num
    AND B_1.sales_amount = max_sales;
```

Sample output

Fname	Mname	Lname	phone_num
William	L.	Briggs	4236619893
Lyle		Cochran	3198049917
Bernard		Gillett	2529551961
Eric		Schulz	3169200013

(f) Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)

```
CREATE VIEW MOST_PROFITABLE_AUTHOR
AS SELECT A_1.Fname, A_1.Mname, A_1.Lname, A_1.phone_num
FROM (SELECT max(B.sales_amount * B.price) AS max_profits
FROM BOOK AS B, AUTHOR AS A, AUTHOR_WRITE_BOOK AS AB
WHERE B.ISBN = AB.ISBN
    AND AB.Aphone_num = A.phone_num) AS R, BOOK AS B_1, AUTHOR AS A_1,
    AUTHOR_WRITE_BOOK AS AB_1
    WHERE B_1.ISBN = AB_1.ISBN
    AND AB_1.Aphone_num = A_1.phone_num
    AND B_1.sales_amount * B_1.price = max_profits;
```

Sample output

Fname	Mname	Lname	phone_num
Bernard		Gillett	2529551961
Eric		Schulz	3169200013
Lyle		Cochran	3198049917
William	L.	Briggs	4236619893

(g) Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.

```
CREATE VIEW CPMPA
AS SELECT DISTINCT C.Fname, C.Lname, C.email
FROM CUSTOMER AS C, TRANSACTION_HISTORY AS T,
BOOK_HAS_TRANSACTION AS BT, AUTHOR_WRITE_BOOK AS AB, MOST_PROFITABLE_AUTHOR
AS MPA
WHERE C.email = T.Cmail
        AND T.transaction_id = BT.transaction_id
        AND BT.ISBN = AB.ISBN
        AND AB.Aphone_num = MPA.phone_num;
```

Sample output

Fname	Lname	email
Elizabeth	White	bplwmkxwpwpnbyi@inbox.com
Brutus	Buckeye	brutusbuckeye@gmail.com
Maria	Martin	ldpcnyrfhqcbljg@inbox.com
Linda	Thomas	qujjcalmqpqkhwm@mail.com
Richard	Miller	tulyybtytrihelv@gmail.com

(h) Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```
CREATE VIEW NEWVIEW_H
AS SELECT DISTINCT A.Fname, A.Mname, A.Lname
FROM more_than_average AS MA, AUTHOR AS A, AUTHOR_WRiTE_BOOK AS AB,
BOOK_HAS_TRANSACTION AS BT, TRANSACTION_HISTORY AS T, CUSTOMER AS C
WHERE MA.Email_Address = T.Cmail
        AND T.transaction_id = BT.transaction_id
        AND BT.ISBN = AB.ISBN
        AND AB.Aphone_num = A.phone_num;
```

Sample output

Fname	Mname	Lname
Harold		McGee
Bernard		Gillett
Eric		Schulz
Lyle		Cochran
William	L.	Briggs
Donald		DeLillo
Mary	Higgins	Clark
Stephen		King
Mark	Z.	Danielewski
Jeffery		Deaver
Terry		Pratchett
Jacquelyn		Mitchard
Patrick		Lencioni
Dee		Henerson
Francis	D.K.	Ching
Cassandra		Adams
Nicholas		Sparks
Stephen		Wolfram
Edward	Osborne	Wilson

7. Description of three sample transactions useful for database.

(1) Add a new book to the database.

```
BEGIN TRANSACTION NEW_BOOK
    INSERT INTO BOOK VALUES ('3334455555','MCGRILL','2');
        IF error THEN GO TO UNDO;END IF;
            INSERT INTO AUTHOR VALUES('JOHN',NULL,'SMITH','614367489');
            IF error THEN GO TO UNDO;END IF;
    COMMIT;
    GO TO FINISH;
    UNDO:
```

```
                ROLLBACK;
        FINISH:
END TRANSACTION
```

(2) Post a review to a book

```
BEGIN TRANSACTION BOOK_REVIEW
        INSERT INTO REVIEW VALUES('21', '143234@gmail.com', '1234567890', '2', NULL);
            IF error THEN GO TO UNDO; END IF;
        COMMIT;
        GO TO FINISH;
  UNDO:
        ROLLBACK;
  FINISH:
  END TRANSACTION
```

(3) A customer buys a new book.

```
BEGIN TRANSACTION BUY-BOOK
INSERT INTO TRANSACTION_HISTORY VALUES('10010', '143234@gmial.com', '2017-05-11', 'TRUE',
NULL, NULL, '2')
          IF error THEN GO TO UNDO;END IF;
          INSERT INTO BOOK_HAS_TRANSACTION VALUES('10010', '1234567890')
              IF error THEN GO TO UNDO; END IF;
INSERT INTO BOOK VALUES('1234567890','OS UP','1','2014-12-12','23.99','Lab Guideline','124')
                  IF error THEN GO TO UNDO; END IF;
      COMMIT;
      GO TO FINISH;
UNDO:
      ROLLBACK;
FINISH:
END TRANSACTION;
```

SECTION 2-USER MANUAL

1. The explanation of each entity, its attributes and constraints of those attribute.

(1)

(a) Entity CUSTOMER_HAS_ADDRESS describes each customer's information.
(b) It has attributes zipcode, street and email. zipcode is provided by each customer, which indicates customer's address information . Street is the street name of a customer. Email is the email of a customer.
(c) Attribute zipcode has datatype of CHAR(5) and constraint of NOT NULL. street has datatype of VARCHAR(30) and constraint of NOT NULL, and email has datatype of VARCHAR(40) and constraint of NOT NULL.
(d) Attributes zipcode,street and email cannot be null. Foreign key street refers to attribute street from table ADDRESS. Foreign key zipcode refers to attribute zipcode from table ADDRESS. Foreign key email refers to attribute email from table CUSTOMER.

(2)
(a) Entity ADDRESS describes each customer's address information.
(b) It has two attributes zipcode and street. zipcode and street are both provided by each customer. zipcode indicates city and state information of the address given by the customer. street indicates which street the customer is living in.
(c) Attribute zipcode has data type CHAR(5). Attribute street has data type VARCHAR(30).
(d) Zipcode and street cannot be null.

(3)
(a) Entity CUSTOMER_HAS_PAYMENT describes payment information of each customer.
(b) It has two attributes email and credit_card. email is the email address of each customer. credit_card is the credit card number provided by each customer.
(c) Attribute email has datatype of VARCHAR(40). Attribute credit_card VARCHAR(19).
(d) Attributes email and credit_card cannot be null. Both email and credit_card are foreign keys from table CUSTOMER and table PAYMENT respectively.

(4)
(a) Entity CUSTOMER describes each customer's basic information - name, phone number and email address.
(b) It has four attributes email, phone_num, Fname and Lname. Fname represents customer's first name, and Lname represents customer's last name. email indicates customers' contact email address. And phone_num indicates customers' contact phone number.
(c) Attribute email has data type VARCHAR(40). Attribute phone_num has data type VARCHAR(20). Attribute Fname has data type VARCHAR(30). And attribute Lname has data type VARCHAR(30).
(d) Attribute email and phone_num cannot be null.

(5)
(a) Entity PAYMENT describes payment information of each customer.
(b) It has four attributes credit_card, expired_date, security_code and zipcode. credit_card is the credit card number provided by each customer. expired_date is the expired date of the credit card of each customer. security_code is the security code of the credit card of each customer. zipcode is provided by each customer which indicates city and state information.

(c) Attribute credit_card has datatype of VARCHAR(19). Attribute expired_date has data type of DATE. Attribute security_code has datatype of VARCHAR(3). Attribute zipcode has datatype of CHAR(5).
(d) Attributes credit_card, expired_date, security_code and zipcode cannot be null. Both street and zipcode are foreign keys from table ADDRESS. Foreign key street refers to attribute street from table ADDRESS. Foreign key zipcode refers to attribute zipcode from table ADDRESS.

(6)
(a) Entity PUBLISHER describes publisher information of each publisher.
(b) It has two attributes name and phone_num. name is the name of each publisher. phone_num is the phone number of each publisher.
(c) Attribute name has data type VARCHAR(100). Attribute phone_num has data type VARCHAR(20).
(d) Attributes name and phone_num cannot be null.

(7)
(a) Entity REVIEW describes reading feedback given by different readers. It contains review contents (if applicable) and the rating of the book given by customers.
(b) It has five attributes r_id, Cmail, ISBN, rating and content. r_id represents review id which is associated with each feedback. Cmail represents customer's' email address. ISBN is the ISBN number associated with each book. rating is the rank of book given by customers. And content, which is optional, is the text feedback of the book given by customers.
(c) Attribute r_id has data type VARCHAR(30). Attribute cmail has data type VARCHAR(40). Attribute ISBN has data type VARCHAR(13). Attribute rating has data type VARCHAR(3). Attribute content has data type VARCHAR(3000).
(d) Except for attribute content, all other attributes cannot be null. Attribute ISBN is a foreign key refer to table BOOK. Attribute cmail is a foreign key which refers to table CUSTOMER.

(8)
(a) Entity INVERNTORY_STORE_BOOK describes the information of each inventory.
(b) It has three attributes ISBN, in_id and amount. ISBN is the ISBN of the book in each inventory. in_id is the inventory of each inventory. amount is the amount book in stock of each inventory.
(c) Attribute ISBN has data type VARCHAR(13). Attribute in_id has data type INT. Attribute amount has data type INT.
(d) Attribute ISBN, in_id and amount cannot be null. Attribute ISBN is a foreign key which refers to table BOOK. Attribute in_id is a foreign key which refers to table INVENTORY.

(9)
(a) Entity INVENTORY describes the information of each inventory.
(b) It has four attributes, inventory_id, capacity, street, zipcode. Inventory_id is the id of each inventory, capacity is the capacity of each inventory, street and zipcode are the street and zipcode information of address of each inventory.
(c) invertory_id has the datatype of INT and cannot be null. capacity has the datatype of INT. street has the datatype of VARCHAR with max length of 30, and it cannot be null. zipcode has the datatype of CHAR

with length of 5, and it cannot be onull. inventory _id is the primary key of this table. The street and zipcode are foreign key from the table address.
(d) Attribute in_id, street and zipcode cannot be null. Attribute street and zipcode are foreign keys which refer to table ADDRESS.

(10)
(a) Entity AUTHOR describes the information of each author.
(b) It has four attributes: Fname, Lname, Mname, and phone_num. Fname is the first name of author. Lname is the last name of author. Mname is the middle of author. phone_ num is the phone number of author.
(c) Lname, Fname, Mname, and phone_num have data type VARCHAR(20).
(d) Lname, Fname, and phone_num cannot be null.

(11)
(a) Entity AUTHOR_WRITE_BOOK describes authors associate with each book.
(b) It has two attributes ISBN and Aphone_num. ISBN is associated with each different book. Aphone_num represents author's phone number.
(c) Attribute ISBN has data type VARCHAR(30). Attribute Aphone_num has data type VARCHAR(20).
(d) Attribute ISBN and Aphone_num cannot be null. Attribute ISBN is a foreign key refers to table BOOK. Attribute Aphone_num is a foreign key which refers to table AUTHOR.

(12)
(a) Entity BOOK describes each book's information.
(b) It has seven attributes: ISBN, Pname, title, price, edition, publish_date and sales_amount. ISBN is associated with each different book. Pname represents publisher's name. Title represents book's title. Price is the price sold by the book. Edition is the book's edition. Publish_date represents the published date of the book. Sales_amount represents the sales amount of the book.
(c) Attribute ISBN has data type VARCHAR(13). Attribute Pname has data type VARCHAR(100). Attribute title has data type VARCHAR(300). Attribute price has data type INT. Attribute edition has data type INT. Attribute publish_date has data type INT. Attribute sales_amount has data type INT.
(d) Attribute ISBN cannot be null. Attribute Pname is a foreign key which refers to table PUBLISHER.

(13)
(a) Entity TRANSACTION_HISTORY describes information about transaction history.

(b) It has seven attributes: transaction_id, Cmail, date, sold, return_date, returned, and amount. transaction_id is the id of transaction. Cmail is customer's email. date is the transaction date. sold is the status of this transaction which can be either true for having sold or false for renting. returned_date is the date of returning rented book. returned is whether borrowers have returned books they borrowed. amount is the number of book for a single transaction.

(c) Attributes sold and returned have data type BOOLEAN. date and returned_date have data type DATE. transaction_id has data type VARCHAR(30). Cmail has data type VARCHAR(40). amount has data type INT.

(d) Cmail and transaction_id cannot be null. Cmail is the foreign key from table CUSTOMER's email. Attribute Cmail is a foreign key which refers to table CUSTOMER.

(14)
(a) Entity BOOK_HAS_TRANSACTION describes relationship between BOOK and TRANSACTION_HISTORY.

(b) It has two attributes: ISBN and transaction_id.
(c) Attribute ISBN has data type VARCHAR(13). Attribute transaction_id has data type VARCHAR(30).
(d) ISBN and transaction_id cannot be null. ISBN is the foreign key from table BOOK's ISBN. transaction_id is the foreign key from table TRANSACTION_HISTORY's transaction_id.
Attribute ISBN is a foreign key which refers to table BOOK. Attribute transaction_id is a foreign key which refers to table TRANSACTION_HISTORY.

(15)
(a) Entity POST_COURSE describes the prerequisite and post course of each course.
(b) It has two attributes: prerequisite_num and post_course_num. Attribute prerequisite_num is the prerequisite of the course. Attribute post_course_num is the course you will study after a course.
(c) Attribute prerequisite has data type VARCHAR(15). Attribute post_course has data type VARCHAR(15).
(d) prerequisite_num and post_course_num cannot be null. prerequisite_num is the foreign key from the table COURSE's Course_num. Attribute prerequisite is a foreign key which refers to table COURSE.

(16)
(a) Entity COURSE_USE_BOOK describes what textbooks are used by each course.
(b) It has two attributes: course_num and ISBN. Attribute course_num is the course id of each course. Attribute ISBN is the ISBN(book id) of each book.
(c) Attribute course_num has data type VARCHAR(15). Attribute ISBN has data type VARCHAR(13).
(d) course_num and ISBN cannot be null. course_num is the foreign key from table COURSE's course_num. ISBN is the foreign key from the table BOOK's ISBN. Attribute course_num is a foreign key which refers to table COURSE. Attribute ISBN is a foreign key which refers to table BOOK.

(17)
(a) Entity BOOK_BELONGS_TO_CATEGORY describe the relations between BOOK and CATEGORY.
(b) It has two attributes which are ISBN and category_name. ISBN is the ISBN of each book. Category_name is the category name of each book.
(c) ISBN is the foreign key from the BOOK table and it has datatype of VARCHAR with maximum length of 13. Category_name is the name of categories, and it is a foreign key from the CATEGORY table. It has the datatype of VARCHAR with maximum length of 30.
(d) ISBN and category_name cannot be null. ISBN could only be sequence of 10 or 13 decimal digits. Attribute category_name is a foreign key which refers to table CATEGORY. Attribute ISBN is a foreign key which refers to table BOOK.

(18)

(a) Entity CATEGORY describes the categories of books.
(b) CATEGORY has only one attribute which is Name. Name is the name of each category.
(c) Name is the name of category. And it has the datatype of VARCHAR with the maximum length of 200.
(d) Name cannot be null.

(19)
(a) Entity COURSE describes the information of courses. And it has only one attribute which is Course_num.
(b) Entity COURSE has one attribute Course_num. Course_num is the course number.
(c) Course_num has the datatype of VARCHAR with maximum length of 15.
(d) Course_num cannot be null.

2. The sample SQL queries that provided in Checkpoints 02 and 03.
(1) Checkpoint#2. Question 3 (Also Checkpoint#3 Question 3)
(a) Find the titles of all books by Pratchett that cost less than $10.
Temp_1 <- BOOK ⋈AUTHOR_WRITE_BOOK.ISBN = BOOK.ISBN(AUTHOR_WRITE_BOOK⋈AUTHOR_WRITE_BOOK.Aphone_num=AUTHOR.phone_num (σLname="Pratchett" AUTHOR))
Result <- π title(σprice<10Temp_1)

```
SELECT Title
FROM BOOK
WHERE Price < 10 and ISBN in
  (SELECT ISBN
  FROM AUTHOR_WRITE_BOOK, AUTHOR
  WHERE AUTHOR_WRITE_BOOK.Aphone_num = AUTHOR.Phone_num
  AND AUTHOR.Lname = 'Pratchett');
```

(b) Give all the titles and their dates of purchase made by a single customer.
Temp_1 <- (BOOK_HAS_TRANSACTION*BOOK)*TRANSACTION_HISTORY
Result <- π title,date(σgmail="brutusbuckeye@gmail.com"(Temp_1))

```
select B.title, T.date
from BOOK B, TRANSACTION_HISTORY T, BOOK_HAS_TRANSACTION BT, CUSTOMER C
where B.ISBN = BT.ISBN AND T.transaction_id = BT.transaction_id AND C.email = T.Cmail and C.email =
'brutusbuckeye@gmail.com';
```

(c) Find the titles and ISBNs for all books with less than 5 copies in stock.
Temp_1 <- ISBN F Sum amount(BOOK*INVENTORY_STORE_BOOK)
Result <- π title,ISBN(σsum_amount<5(Temp_1))

```
SELECT  B.title, B.ISBN
```

```
FROM BOOK AS B, INVENTORY_STORE_BOOK AS ISB
WHERE ISB.amount<5 AND ISB.ISBN = B.ISBN;
```

(d) Give all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased.

Temp_1 <-
CUSTOMER⋈CUSTOMER.email=TRANSACTION_HISTORY.Cmail(TRANSACTION_HISTORY⋈TRANSACTION_HISTORY.transaction_id=BOOK_HAS_TRANSACTION.transaction_id(BOOK_HAS_TRANSACTION))
Temp_2 <-
Temp_1⋈TEMP_1.ISBN=AUTHOR_WRITE_BOOK.ISBN(AUTHOR_WRITE_BOOK⋈AUTHOR_WRITE_BOOK.Aphone_num=AUTHOR.phone_num(AUTHOR))
Result <- π Fname,Lname,title (σTemp_2.Lname='Pratchett'(Temp_2))

```
SELECT  C.Fname, C.Lname, B.title
FROM CUSTOMER C, BOOK B, AUTHOR A, AUTHOR_WRITE_BOOK AWB, TRANSACTION_HISTORY T,
BOOK_HAS_TRANSACTION BT
WHERE  A.phone_num = AWB.Aphone_num
                AND AWB.ISBN = B.ISBN
                AND B.ISBN = BT.ISBN
                AND T.Cmail = C.email
                AND BT.transaction_id = T.transaction_id
                AND  A.Lname  =  'Pratchett';
```

(e) Find the total number of books purchased by a single customer

Temp_1 <- σemail="brutusbuckeye@gmail.com"AND sold="true"(TRANSACTION_HISTORY)
Result <- F sum amount(Temp_1)

```
SELECT SUM(T.amount)
FROM CUSTOMER C, TRANSACTION_HISTORY T
WHERE T.sold = 'TRUE'
        AND C.email = T.Cmail
        AND C.Fname = 'John' AND C.Lname = 'Smith';
```

(f) Find the customer who has purchased the most books and the total number of books they have purchased.

Temp_1 <- email F count amount (σsold="true"(TRANSACTION_HISTORY))
Result <- email F max sum amount(Temp_1)

```
select Lname, Fname, Max(s_amount)
from (select C.Lname, C.Fname, sum(T.amount) as s_amount
        from CUSTOMER AS C, TRANSACTION_HISTORY AS T
        where T.sold = 'TRUE' and C.email = T.Cmail
        group by C.email);
```

(2) Checkpoint#2. Question 4 (Also Checkpoint#3. Question4)

(a) Find titles and ratings of books which have the highest average rating.

Temp_1 <- ISBN F max rating(REVIEW)
Result <- π title,max_rating(Temp_1*BOOK)

```
SELECT title, max(avg_rating)
FROM
   (SELECT title, avg(Rating) AS avg_rating
    FROM BOOK, REVIEW
    WHERE BOOK.ISBN = REVIEW.ISBN
    GROUP BY BOOK.ISBN);
```

* Assume a customer bought textbooks for himself or herself

(b) Find all the courses he or she takes this term

Temp_1 <- σ Cmail = ”brutusbuckeye@gmail.com”(TRANSACTION_HISTORY)
Result <- π course_num (BOOK_HAS_TRANSACTION * Temp_1)*COURSE_USE_BOOK

```
SELECT C.email, CUB.course_num
FROM COURSE_USE_BOOK AS CUB, TRANSACTION_HISTORY AS T, BOOK AS B, CUSTOMER AS C,
BOOK_HAS_TRANSACTION AS BHT
WHERE T.Cmail = 'brutusbuckeye@gmail.com'
   AND C.email = T.Cmail
   AND T.transaction_id = BHT.transaction_id
   AND BHT.ISBN = B.ISBN
   AND B.ISBN = CUB.ISBN
```

(c) Find all textbooks he or she might want to buy in next term

Temp_1 <- σ Cmail=”brutusbuckeye@gmail.com”(TRANSACTION_HISTORY)
Temp_2 <- π course_num (BOOK_HAS_TRANSACTION * Temp_1) * COURSE_USE_BOOK
Temp_3 <- Temp_2 ⋈Temp_2.course_num = POST_COURSE.prerequisite_num (POST_COURSE)
Result <- π ISBN(Temp_3 ⋈Temp_3.post_course_num= COURSE_USE_BOOK.course_num (COURSE_USE_BOOK)

```
SELECT T.Cmail, B.title
FROM TRANSACTION_HISTORY T, BOOK_HAS_TRANSACTION BT, COURSE_USE_BOOK CB, POST_COURSE
PC, BOOK B
WHERE T.Cmail = 'brutusbuckeye@gmail.com'
   AND BT.transaction_id = T.transaction_id
   AND CB.ISBN = BT.ISBN
   AND PC.prerequisite_num = CB.course_num
   AND B.ISBN = BT.ISBN
```

(3) Checkpoint#3. Question 5

(a) Provide a list of customer names, along with the total dollar amount each customer has spent.

```
CREATE VIEW money_spent(Fist_name, Last_name, dollar_spent)
AS SELECT C.Fname, C.Lname, sum(T.amount * B.price)
FROM CUSTOMER C, BOOK B, TRANSACTION_HISTORY T, BOOK_HAS_TRANSACTION BT
WHERE C.email = T.Cmail
        AND T.transaction_id = BT.transaction_id AND B.ISBN = BT.ISBN
GROUP BY C.email
```

(b) Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.

```
CREATE VIEW more_than_average(First_name, Last_name, Email_Address)
        AS SELECT Fname, Lname, email
FROM (SELECT Fname, Lname, email, sum(T.amount * B.price) as cost
FROM CUSTOMER C, BOOK B, TRANSACTION_HISTORY T, BOOK_HAS_TRANSACTION BT
WHERE C.email = T.Cmail AND T.transaction_id = BT.transaction_id AND B.ISBN = BT.ISBN
GROUP BY C.email),
(SELECT avg(dollar_spent) AS avg_spent
        FROM money_spent)
WHERE  cost > avg_spent;
```

(c) Provide a list of the titles in the database and associated total copies sold to customers, sorted from the title that has sold the most individual copies to the title that has sold the least.

```
CREATE VIEW BEST_SELL
AS SELECT B.title, SUM(T.amount) AS sales
FROM BOOK AS B, TRANSACTION_HISTORY AS T,BOOK_has_TRANSACTION AS BHT
WHERE T.sold = 'TRUE'
        AND T.transaction_id = BHT.transaction_id
        AND BHT.ISBN = B.ISBN
GROUP BY BHT.ISBN
ORDER BY sales DESC;
```

(d) Provide a list of the titles in the database and associated dollar totals for copies sold to customers, sorted from the title that has sold the highest dollar amount to the title that has sold the smallest.

```
CREATE VIEW BEST_PROFITS
```

```
AS SELECT B.title, sum(T.amount * B.price) AS profits
FROM BOOK AS B, TRANSACTION_HISTORY AS T, BOOK_has_TRANSACTION AS BT
WHERE T.sold = 'TRUE'
        AND T.transaction_id = BT. transaction_id
        AND BT.ISBN = B.ISBN
GROUP BY BT.ISBN
ORDER BY profits DESC;
```

(e) Find the most popular author in the database (i.e. the one who has sold the most books)

```
CREATE VIEW MOST_POPULAR_AUTHOR
AS SELECT A_1.Fname, A_1.Mname, A_1.Lname, A_1.phone_num
FROM (SELECT max(B.sales_amount) AS max_sales
FROM BOOK AS B, AUTHOR AS A, AUTHOR_WRITE_BOOK AS AB
WHERE B.ISBN = AB.ISBN
AND AB.Aphone_num = A.phone_num) AS R, BOOK AS B_1, AUTHOR AS A_1,
AUTHOR_WRITE_BOOK AS AB_1
WHERE B_1.ISBN = AB_1.ISBN
        AND AB_1.Aphone_num = A_1.phone_num
        AND B_1.sales_amount = max_sales;
```

(f) Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)

```
CREATE VIEW MOST_PROFITABLE_AUTHOR
AS SELECT A_1.Fname, A_1.Mname, A_1.Lname, A_1.phone_num
FROM (SELECT max(B.sales_amount * B.price) AS max_profits
FROM BOOK AS B, AUTHOR AS A, AUTHOR_WRITE_BOOK AS AB
WHERE B.ISBN = AB.ISBN
        AND AB.Aphone_num = A.phone_num) AS R, BOOK AS B_1, AUTHOR AS A_1,
        AUTHOR_WRITE_BOOK AS AB_1
        WHERE B_1.ISBN = AB_1.ISBN
        AND AB_1.Aphone_num = A_1.phone_num
        AND B_1.sales_amount * B_1.price = max_profits;
```

(g) Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.

```
CREATE VIEW CPMPA
```

```
AS SELECT DISTINCT C.Fname, C.Lname, C.email
FROM CUSTOMER AS C, TRANSACTION_HISTORY AS T,
BOOK_HAS_TRANSACTION AS BT, AUTHOR_WRITE_BOOK AS AB, MOST_PROFITABLE_AUTHOR
AS MPA
WHERE C.email = T.Cmail
        AND T.transaction_id = BT.transaction_id
        AND BT.ISBN = AB.ISBN
        AND AB.Aphone_num = MPA.phone_num;
```

(h) Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.

```
CREATE VIEW NEWVIEW_H
AS SELECT DISTINCT A.Fname, A.Mname, A.Lname
FROM more_than_average AS MA, AUTHOR AS A, AUTHOR_WRiTE_BOOK AS AB,
BOOK_HAS_TRANSACTION AS BT, TRANSACTION_HISTORY AS T, CUSTOMER AS C
WHERE MA.Email_Address = T.Cmail
        AND T.transaction_id = BT.transaction_id
        AND BT.ISBN = AB.ISBN
        AND AB.Aphone_num = A.phone_num;
```

2. INSERT syntax for adding new books, publishers, authors and customers to the system.

```
(a) INSERT INTO BOOK VALUES ('3334455555','MCGRILL','2');
(b) INSERT INTO PUBLISHER VALUES ('Del Rey', '1234567890');
(c) INSERT INTO AUTHOR VALUES('JOHN',NULL,'SMITH','614367489');
(d) INSERT INTO CUSTOMER VALUES('1232345@gmail.com', '1928304812', 'Chloe', 'Chen');
```

3. DELETE syntax for removing books, publishers, authors and customers from the BookStore system.

```
(a) DELETE FROM BOOK
   WHERE ISBN = '3334455555' AND Pname = 'MCGRILL' AND edition = '2';
(b) DELETE FROM PUBLISHER
   WHERE name = 'Del Rey' AND phone_num = '1234567890';
(c) DELETE FROM AUTHOR
   WHERE Fname = 'JOHN' AND Lname = 'SMITH';
(d) DELETE FROM CUSTOMER
    WHERE email = '1232345@gmail.com' AND phone_num = '1928304812' AND
    Fname='Chloe' AND Lname='Chen';
```

Section 3 - Graded Checkpoints Documents

1. Graded Checkpoints

Please refer to the appendix file appendix 7 to 10

2. Original checkpoints

Please refer to the appendix file appendix 3 to 6

3. Revised version of checkpoint documents

(1) Checkpoint 1

❖ Question#1 See below

1.book: published date, ISBN, edition, category(name), price, title, sales_amount, in_stock_amount.
2.author: name(first_name, last_name), phone_number
3.publisher: name, phone_number
4.customer: name, email, phone number, credit_card_num,
5.address(street, zipcode)

❖ Question#3 See below

1.transaction_history: transaction_id, status(is_Sold/ is_Rented(dates(rent_date, return_data), returned))
2.course: course_number, post_course_num

CUSTOMER entities generate TRANSACTION_HISTORY entities.
COURSE entities post COURSE entities.
COURSE entities use BOOK entities.
TRANSACTION_HISTORY entity is related to BOOK_HAS_TRANSACTION entity.
BOOK_HAS_TRANSACTION entity is related to BOOK entity.
BOOK entity is related to COURSE_USE_BOOK entity.
COURSE_USE_BOOK entity is related to COURSE entity

(2) Checkpoint 2

❖ Question#3. See Section 1 Database Description (Page 11 to 13) for the revised relational algebra to perform required queries.

❖ Question#4. See Section 1 Database Description (Page 12 to 13) for the revised relational algebra to perform additional interesting queries.

(3) Checkpoint 3

❖ Question#3. See Page 11 to 12 for the revised queries to "WorksheetTwoSimpleQueries.txt".

❖ Question#4. See Page 12 to 13 for the revised queries to "WorksheetTwoExtraQueries.txt".

❖ Question#5. See Page 13 to 15 for the revised queries to "WorksheetTwoAdvancedQueries.txt".

(4) Checkpoint 4

❖ Question#2. See section DB Description Page 3 to 4 for the revised functional dependencies.

- Question#5. See section DB Description Page 6 for the revised views built from our database.