Graded Checkpoint Documents

CSE 3241 Project Checkpoint 01 – Entities and Relationships

Group members: Tingwei Duan, Chen Cai, Hui Li, En-Ju Lin

1. Based on the requirements given in the project overview, list the entities to be modeled in this database. For each entity, provide a list of associated attributes.

BOOK: ISBN, Title, Year, Price, Language, Format
CUSTOMER: Name, Telephone Number, Home Address, Email address

ADMINISTRATOR: Name, ID
PUBLISHER: Name, Home Address, Email Address, Telephone Number
ORDER: Order_ID, Status, Order_Info
CART: ID, Quantity, Total Price
INVENTORY: ISBN, Quantity in store, Quantity on orders.
PAYMENT: Name on card, Card Number, Card Type, Expiry Date
AUTHOR: First_Name, MI, Last_Name

2. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, "CUSTOMER entities purchase BOOK entities").

CART can have many BOOK. BOOK can go to many CART (Assume: book is not a specific item).

ORDER can only be created by one CART. CART can create only one ORDER.

PAYMENT should be received per ORDER. ORDER can receive only one PAYMENT.

CUSTOMER can rate many BOOK. BOOK can be rated by many CUSTOMER.

PUBLISHER can publish many BOOK. BOOK can be published by only one PUBLISHER.

INVENTORY can consists many BOOK. BOOK can be in many INVENTORY (Assume: book is the collection of same product).

ADMINISTRATOR can order many books from PUBLISHER. PUBLISHER must receive order from at least one ADMINISTRATOR.

ADMINISTRATOR can manage at least one to many INVENTORY. INVENTORY is managed by at least one to many ADMINISTRATOR.

BOOK must have at least one AUTHORS. AUTHORS can write many BOOK.

3. Propose at least two additional entities that it would be useful for this database to model beyond the scope of the project requirements. Provide a list of possible attributes for the additional entities and possible relationships they may have with each other and the rest of the entities in the database. Give a brief, one sentence rationale for why adding these entities would be interesting/useful to the stakeholders for this database project.

We propose to add "WAREHOUSE" and "FEATURES" as additional entities as described below.

- WAREHOUSE (city, state, address, phone number)

  Relationships with other entities: Each WAREHOUSE holds one INVENTORY; INVENTORY can be stored in multiple WAREHOUSE.

  Rationale: Can handle inventory and distribution from multiple warehouses

- CATEGORY (Category_ID, Genre)

  Relationships with other entities: BOOK has many features CATEGORY. CATEGORY can belong to many BOOK.

  Rationale: Facilitates customers to search books by book category, e.g. fiction, comic, travel and etc.

4. Give at least four examples of some informal queries/reports that it might be useful for this database might be used to generate. Include one example for each of the additional entities you proposed in question 3 above.

- What are the best-selling books over the past month (or other time period)?

  Queries on WAREHOUSE

- Where is the closest warehouse to process this order?
- How many orders do each warehouse process this month (or any other time period of interest)?

  Queries on CATEGORY

- Reports by administrator may include: What is the best selling category?

- Query by customers include: List all books on biography.

5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you've outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your model to allow for publishers to be added as separate entities.

Yes, based on our above description, we already made PUBLISHER as a separate entity. The new PUBLISHER entity has Name, Address, Email Address, Telephone Number. The PUBLISHER is not an attribute of the BOOK entity. BOOK is published by PUBLISHER. ADMINISTRATOR orders from PUBLISHER.

6. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 3 above.

- Informal update operation: Add another warehouse
  Entities and attributes changed: all attributes related to WAREHOUSE will be changed (i.e code, address and phone).

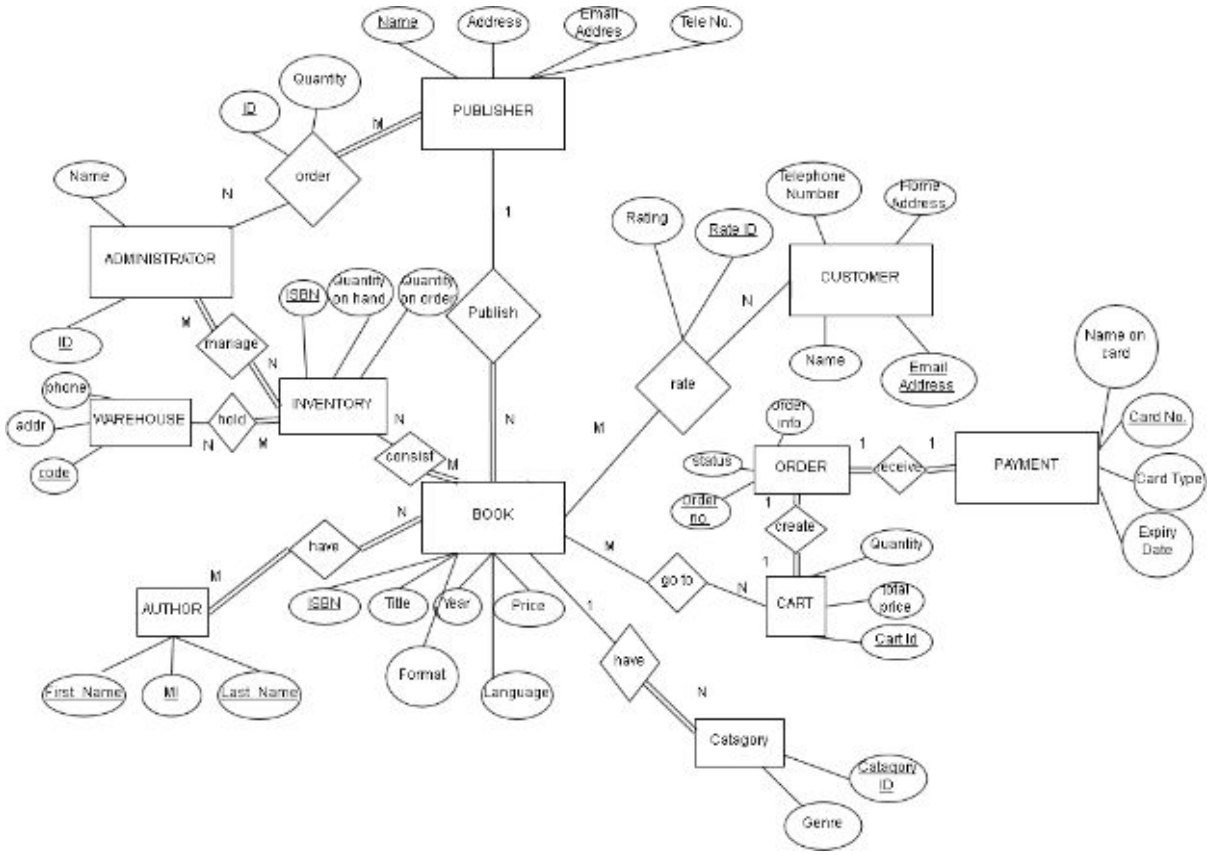
- Informal update operation: Change warehouse location
  Entities and attributes changed: WAREHOUSE (address).

- Informal update operation: Change publisher telephone number
  Entities and attributes changed: PUBLISHER (telephone number)

- Informal update operation: Change "Mathematics" to "Maths" in CATEGORY
  Entities and attributes changed: CATEGORY (Genre)

- Informal update operation: Change customer telephone number
  Entities and attributes changed: CUSTOMER (telephone number)

7. Provide an ER diagram for your database. Make sure you include all of the entities and relationships you determined in the questions above *INCLUDING the entities for question 3 above*, and remember that *EVERY* entity in your model needs to connect to another entity in the model via some kind of relationship.



CSE 3241 Project Checkpoint 01_Individual Feedback

Q4. Where is the closest warehouse to process this order? You cannot get the report/query about it from database.(See Checkpoint 01_Revised for the new relational model diagram)

Q7.

1. Customer not able to make order. You have to add relationship between customer and order.

2. Customer should relate to the payment in order to track who made the payment. Furthermore, you use the card number as the unique attributes. But a customer can use one card to make payment to several orders. Make it a transaction and have a transaction id or change the cardinality.

3. Since an order created exact one cart and the attribute of cart is also check the order. It seems not make any sense to have a cart entity. You can simply ERD by delete the cart entity and make total price as an attribute to the order and quantity as an attribute to the relationship between book and order.

4. Since you use the ISBN as a key attribute to the inventory, you can simpely it by making inventory/quantity as an attribute to the store_in(relationship between book and warehouse)
(See Checkpoint 01_Revised for the new relational model diagram)

CSE 3241 CheckPoint 1_Revised

Group members: Tingwei Duan, Chen Cai, Hui Li, En-Ju Lin

1. Based on the requirements given in the project overview, list the entities to be modeled in this database. For each entity, provide a list of associated attributes.

BOOK: ISBN, Title, Year, Price, Language, Format
CUSTOMER: Name, Telephone Number, Home Address, Email address

ADMINISTRATOR: Name, ID
PUBLISHER: Name, Home Address, Email Address, Telephone Number
ORDER: Order_ID, Status, Order_Info
CART: ID, Quantity, Total Price
INVENTORY: ISBN, Quantity in store, Quantity on orders.
PAYMENT: Name on card, Card Number, Card Type, Expiry Date
AUTHOR: First_Name, MI, Last_Name

2. Based on the requirements given in the project overview, what are the various relationships between entities? (For example, "CUSTOMER entities purchase BOOK entities").

CART can have many BOOK. BOOK can go to many CART (Assume: book is not a specific item).

ORDER can only be created by one CART. CART can create only one ORDER.

PAYMENT should be received per ORDER. ORDER can receive only one PAYMENT.

CUSTOMER can rate many BOOK. BOOK can be rated by many CUSTOMER.

PUBLISHER can publish many BOOK. BOOK can be published by only one PUBLISHER.

INVENTORY can consists many BOOK. BOOK can be in many INVENTORY (Assume: book is the collection of same product).

ADMINISTRATOR can order many books from PUBLISHER. PUBLISHER must receive order from at least one ADMINISTRATOR.

ADMINISTRATOR can manage at least one to many INVENTORY. INVENTORY is managed by at least one to many ADMINISTRATOR.

BOOK must have at least one AUTHORS. AUTHORS can write many BOOK.

3. Propose at least two additional entities that it would be useful for this database to model beyond the scope of the project requirements. Provide a list of possible attributes for the additional entities and possible relationships they may have with each other and the rest of the entities in the database. Give a brief, one sentence rationale for why adding these entities would be interesting/useful to the stakeholders for this database project.

We propose to add “WAREHOUSE” and “FEATURES” as additional entities as described below.

- WAREHOUSE (city, state, address, phone number)

  Relationships with other entities: Each WAREHOUSE holds one INVENTORY; INVENTORY can be stored in multiple WAREHOUSE.

  Rationale: Can handle inventory and distribution from multiple warehouses

- CATEGORY (Category_ID, Genre)

  Relationships with other entities: BOOK has many features CATEGORY. CATEGORY can belong to many BOOK.

  Rationale: Facilitates customers to search books by book category, e.g. fiction, comic, travel and etc.

4. Give at least four examples of some informal queries/reports that it might be useful for this database might be used to generate. Include one example for each of the additional entities you proposed in question 3 above.

- What are the best-selling books over the past month (or other time period)?

  Queries on WAREHOUSE

- How many orders do each warehouse process this month (or any other time period of interest)?

Queries on CATEGORY

- Reports by administrator may include: What is the best selling category?
- Query by customers include: List all books on biography.

5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you've outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your model to allow for publishers to be added as separate entities.

Yes, based on our above description, we already made PUBLISHER as a separate entity. The new PUBLISHER entity has Name, Address, Email Address, Telephone Number. The PUBLISHER is not an attribute of the BOOK entity. BOOK is published by PUBLISHER. ADMINISTRATOR orders from PUBLISHER.

6. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 3 above.

- Informal update operation: Add another warehouse
  Entities and attributes changed: all attributes related to WAREHOUSE will be changed (i.e code, address and phone).

- Informal update operation: Change warehouse location
  Entities and attributes changed: WAREHOUSE (address).

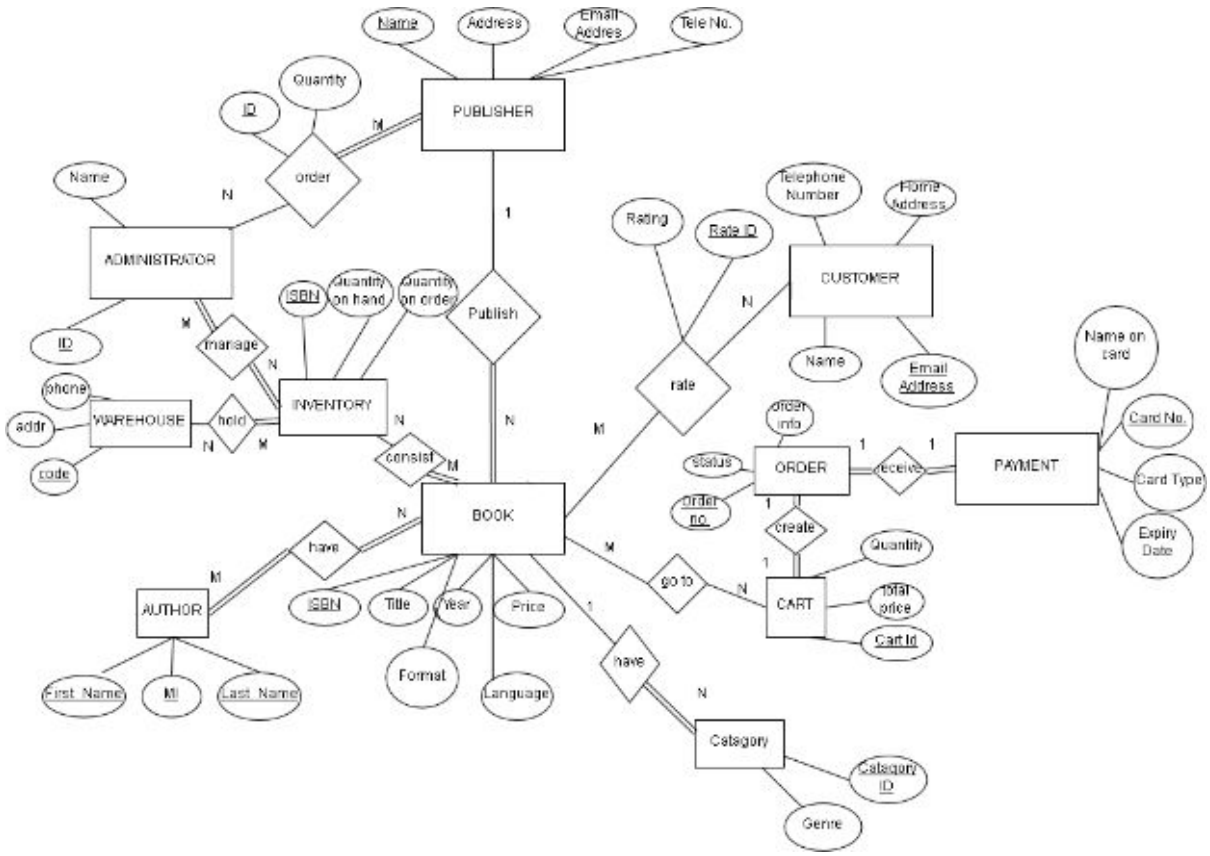
- Informal update operation: Change publisher telephone number
  Entities and attributes changed: PUBLISHER (telephone number)

- Informal update operation: Change "Mathematics" to "Maths" in CATEGORY
  Entities and attributes changed: CATEGORY (Genre)

- Informal update operation: Change customer telephone number
  Entities and attributes changed: CUSTOMER (telephone number)

7. Provide an ER diagram for your database. Make sure you include all of the entities and relationships you determined in the questions above *INCLUDING the entities for question 3 above*, and remember that *EVERY* entity

in your model needs to connect to another entity in the model via some kind of relationship.



CSE 3241 Project Checkpoint 02 – Relational Model and Relational Algebra

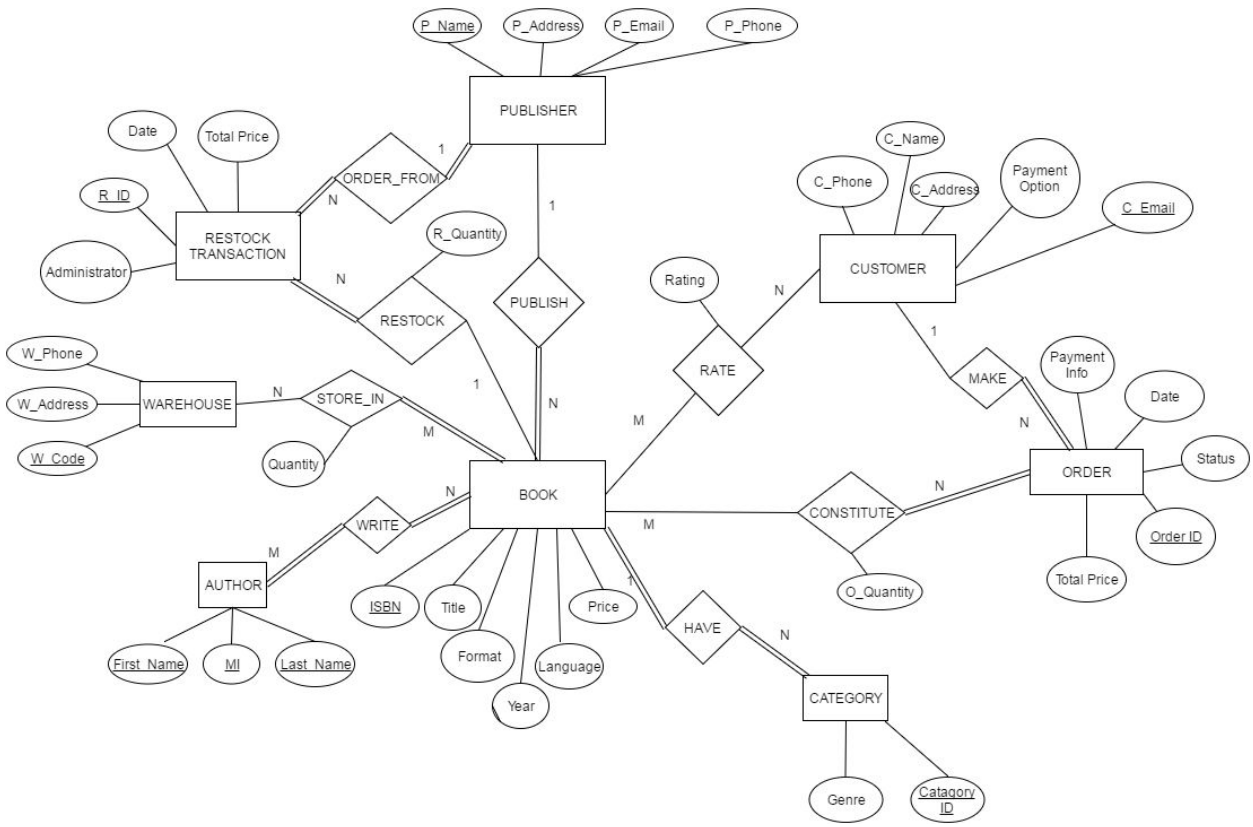
Names: Chen Cai Date 06/03/2016

Tingwei Duan

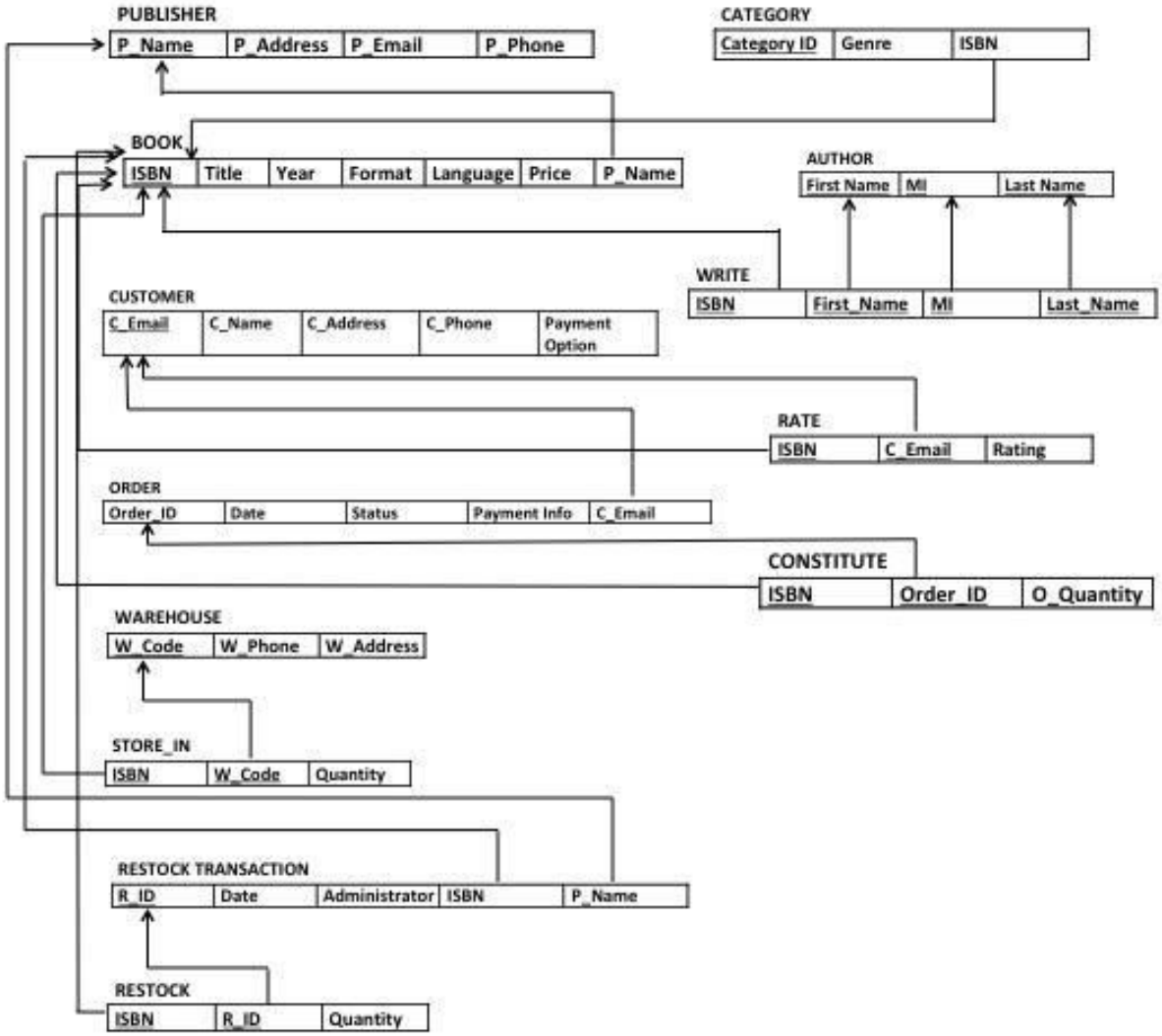
Hui Li

En-Ju Lin

1. Provide a current version of your ER Model as per Project Checkpoint 01. If you were instructed to change the model for Project Checkpoint 01, make sure you use the revised version of your ER Model.



2. Map your ER model to a relational schema. Indicate all primary and foreign keys.



3. Given your relational schema, provide the relational algebra to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries:

    a. Find the titles of all books by Pratchett that cost less than $10

$$R\_BOOKS \leftarrow \delta_{\text{Last Name = Pratchett}}(\text{WRITE})$$
$$BOOKS \leftarrow (R\_BOOKS \bowtie_{\text{R\_BOOKS.ISBN = BOOK.ISBN}} \text{BOOK})$$
$$R1 \leftarrow (\delta_{\text{Price < \$ 10}}(\text{BOOKS}))$$
$$Result \leftarrow \pi_{\text{<Title>}}(R1)$$

b. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

Since customer is specified by email. We assume the single customer is defined by “buck.1@osu.edu”

$$BUCK \leftarrow (\delta_{email=buck.1@osu.edu}(CUSTOMER))$$
$$T1 \leftarrow (BUCK \bowtie_{BUCK.Email = Order.Email} ORDER)$$
$$Result \leftarrow \pi_{<Title,Date>}(T1)$$

c. Find the titles and ISBNs for all books with less than 5 copies in stock

$$BOOK\_QUAN \leftarrow \rho_{R(ISBN, TotalQuan)}(_{ISBN}\mathcal{F}_{SUM\ Quantity}(STORE\_IN))$$
$$BOOK\_LESSTHAN5 \leftarrow \delta_{TotalQuan<5}(BOOK\_QUAN)$$
$$RESULT \leftarrow \pi_{ISBN, Title}(BOOK\_LESSTHAN5 \bowtie_{BOOK\_LESSTHAN5.ISBN = BOOK.ISBN} BOOK)$$

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

$$R\_BOOKS \leftarrow \delta_{Last\ Name = Pratchett}(WRITE)$$
$$BOOKS \leftarrow (R\_BOOKS \bowtie_{R\_BOOKS.ISBN = BOOK.ISBN} BOOK)$$
$$T1 \leftarrow (BOOKS \bowtie_{BOOKS.ISBN = CONSTITUTE.ISBN} CONSTITUTE)$$
$$T2 \leftarrow (T1 \bowtie_{T1.Order\_ID = ORDER.Order\_ID} ORDER)$$
$$RES \leftarrow (CUSTOMER \bowtie_{CUSTOMER.Email = T2.Email} T2)$$

e. Find the total number of books purchased by a single customer (you choose how to designate the customer)

Since customer is specified by email address. We assume the single customer is defined by “buck.1@osu.edu”

$$BUCK \leftarrow \delta_{Email=buck.1@osu.edu}(CUSTOMER)$$
$$B\_ORDER \leftarrow (BUCK \bowtie_{BUCK.Email = ORDER.Email} ORDER)$$
$$TEMP \leftarrow (B\_ORDER \bowtie_{B\_ORDER.Order\_ID = CONSTITUTE.Order\_ID} CONSTITUTE)$$
$$RES \leftarrow \mathcal{F}_{SUM\ Quantity}(Temp)$$

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

$ALL\_ORDER \leftarrow (CUSTOMER \bowtie_{CUSTOMER.Email = ORDER.Email} ORDER)$

$MATCH\_CONS \leftarrow (ALL\_ORDER \bowtie_{ALL\_ORDER.Order\_ID = CONSTITUTE.Order\_ID} CONSTITUTE)$

$RES1 \leftarrow {}_{Email}\mathcal{F}_{SUM\ Quantity}(MATCH\_CONS)$

$RES \leftarrow \mathcal{F}_{MAX\ Quantity}(RES1)$

4. Come up with three additional interesting queries that your database can provide. Give what the queries are supposed to retrieve in plain English and then as relational algebra. Your queries should include joins and at least one should include an aggregate function. At least one of your queries should use "extra" entities you added to your model in Checkpoint 01.

Queries on WAREHOUSE(From Checkpoint 01 extra entity)

- List all book titles in the W_Code is 100.

$\pi_{(Title)}(\ BOOK \bowtie_{(BOOK.ISBN=STORE\_IN.ISBN)} ((\delta_{(W\_code=100)}(STORE\_IN))))$

Queries on CATEGORY(From Checkpoint 01 extra entity)

- List all books on biography.

$\pi_{(Title)}(\ BOOK \bowtie_{(BOOK.ISBN=CATEGORY.ISBN)} ((\delta_{(Genre = "Biography")}\ CATEGORY)))$

Queries on ORDER

- Compute the number of order and average total price in ORDER entity.

$\mathcal{F}_{COUNT(Order\_ID),AVERAGE(Total\ Price)}(ORDER)$

CSE 3241 Project Checkpoint 02_Individual Feedback

Q1. (See Checkpoint 02_Revised for the new relational model diagram)
·There may be multiple authors with the same name. Create a new AuthorID as PK.
·Customers may want to store multiple addresses, so make ADDRESS entity. You can also use this for PUBLISHER and WAREHOUSE.
·It may be useful to separate the parts of an address(state, zip code, country). Then you can search for all orders in Ohio, for example.
·A CATEGORY may have multiple books in it.
·Use first_name, last_name, middle_initial. More standardized.
Q2. (See Checkpoint 02_Revised for the new schema)

·Nice mapping! Just remember when mapping 1:N relationships with attributes, the attributes of the relationship are added as attributes to the entity on the N side of the relationship. (Delete RESTOCK and add 'Quantity' to RESTOCK_TRANSACTION.)
Q3&4. (See Checkpoint 02_Revised for the new relational algebras)
·Remember to project(π) the required attributes, and make sure you don't refer to attributes that aren't present in your current table.

CSE 3241 Project Checkpoint 02_Revised

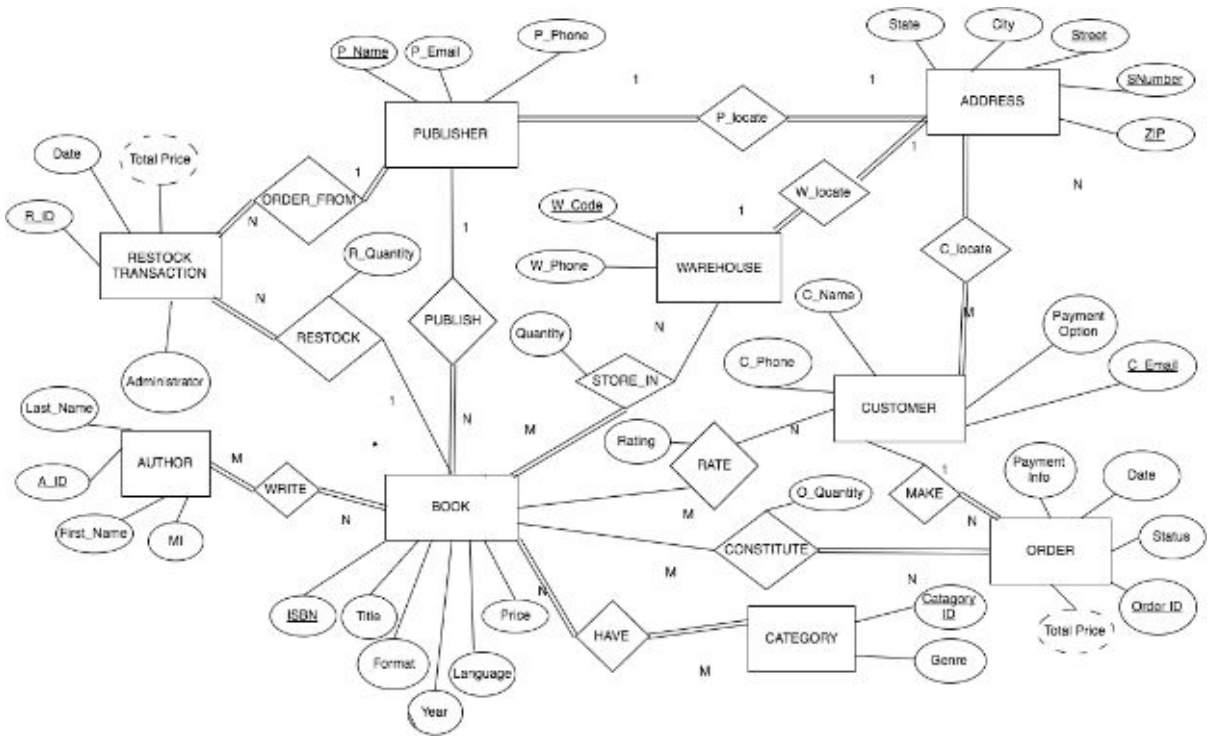
Names: Chen Cai Date 06/03/2016

Tingwei Duan

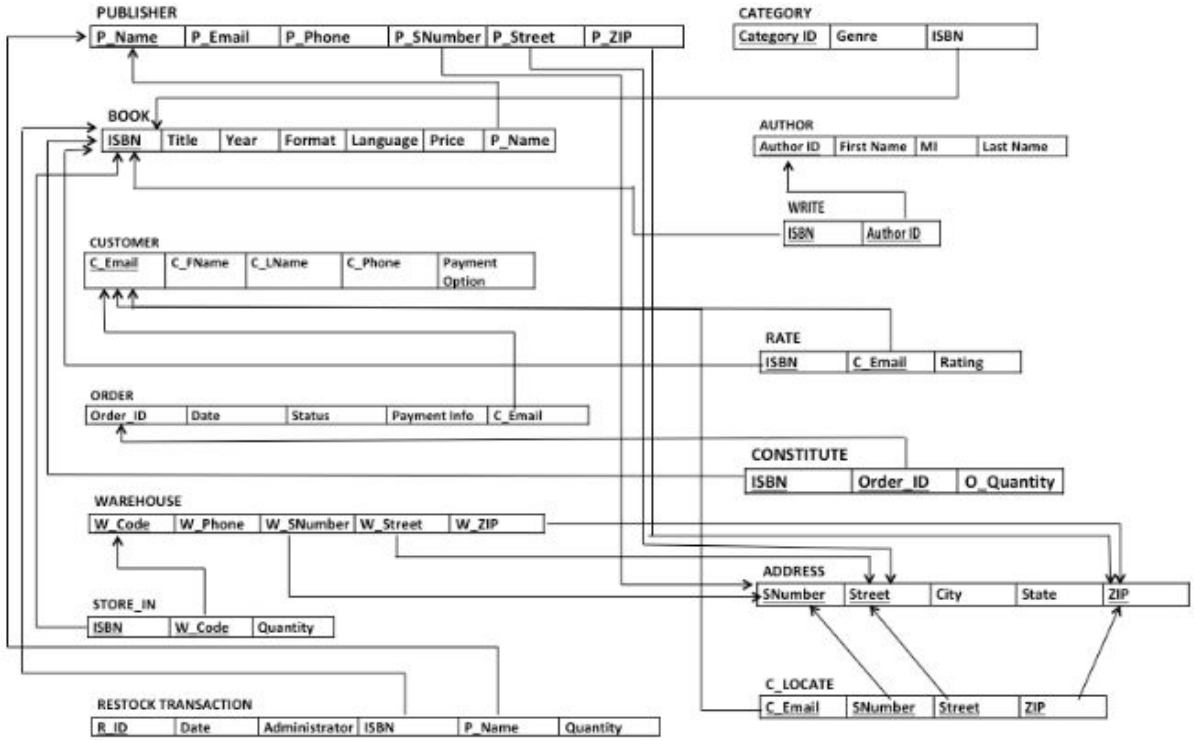
Hui Li

En-Ju Lin

1. Provide a current version of your ER Model as per Project Checkpoint 01. If you were instructed to change the model for Project Checkpoint 01, make sure you use the revised version of your ER Model.



2. Map your ER model to a relational schema. Indicate all primary and foreign keys.



3. Given your relational schema, provide the relational algebra to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries:

a. Find the titles of all books by Pratchett that cost less than $10

$$R\_BOOKS \leftarrow \delta_{\text{Last Name = Pratchett}}(\text{WRITE})$$
$$BOOKS \leftarrow (\ R\_BOOKS \bowtie_{\text{R\_BOOKS.ISBN = BOOK.ISBN}} \text{BOOK}\ )$$
$$R1 \leftarrow (\delta_{\text{Price < \$ 10}}(\text{BOOKS}))$$
$$Result \leftarrow \pi_{\text{<Title>}}(\text{R1})$$

b. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

Since customer is specified by email. We assume the single customer is defined by “buck.1@osu.edu”

$$BUCK \leftarrow (\delta_{\text{email=buck.1@osu.edu}}(\text{CUSTOMER}))$$

$$T1 \leftarrow (BUCK \bowtie_{BUCK.Email = Order.Email} ORDER)$$
$$Result \leftarrow \pi_{<Title,Date>}(T1)$$

c. Find the titles and ISBNs for all books with less than 5 copies in stock

$$BOOK\_QUAN \leftarrow \rho_{R(ISBN,\ TotalQuan)}({}_{ISBN}\mathcal{F}_{SUM\ Quantity}(STORE\_IN))$$
$$BOOK\_LESSTHAN5 \leftarrow \delta_{TotalQuan<5}(BOOK\_QUAN)$$
$$RESULT \leftarrow \pi_{ISBN,\ Title}(BOOK\_LESSTHAN5 \bowtie_{BOOK\_LESSTHAN5.ISBN\ =\ BOOK.ISBN} BOOK)$$

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

$$R\_BOOKS \leftarrow \delta_{Last\ Name\ =\ Pratchett}(WRITE)$$
$$BOOKS \leftarrow (R\_BOOKS \bowtie_{R\_BOOKS.ISBN\ =\ BOOK.ISBN} BOOK)$$
$$T1 \leftarrow (BOOKS \bowtie_{BOOKS.ISBN\ =\ CONSTITUTE.ISBN} CONSTITUTE)$$
$$T2 \leftarrow (T1 \bowtie_{T1.Order\_ID\ =\ ORDER.Order\_ID} ORDER)$$
$$RES \leftarrow (CUSTOMER \bowtie_{CUSTOMER.Email\ =\ T2.Email} T2)$$

e. Find the total number of books purchased by a single customer (you choose how to designate the customer)

Since customer is specified by email address. We assume the single customer is defined by "buck.1@osu.edu"

$$BUCK \leftarrow \delta_{Email=buck.1@osu.edu}(CUSTOMER)$$
$$B\_ORDER \leftarrow (BUCK \bowtie_{BUCK.Email\ =\ ORDER.Email} ORDER)$$
$$TEMP \leftarrow (B\_ORDER \bowtie_{B\_ORDER.Order\_ID\ =\ CONSTITUTE.Order\_ID} CONSTITUTE)$$
$$RES \leftarrow \mathcal{F}_{SUM\ Quantity}(Temp)$$

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

$$ALL\_ORDER \leftarrow (CUSTOMER \bowtie_{CUSTOMER.Email\ =\ ORDER.Email} ORDER)$$
$$MATCH\_CONS \leftarrow (ALL\_ORDER \bowtie_{ALL\_ORDER.Order\_ID\ =\ CONSTITUTE.Order\_ID} CONSTITUTE)$$
$$RES1 \leftarrow {}_{Email}\mathcal{F}_{SUM\ Quantity}(MATCH\_CONS)$$
$$RES \leftarrow \mathcal{F}_{MAX\ Quantity}(RES1)$$

5. Come up with three additional interesting queries that your database can provide. Give what the queries are supposed to retrieve in plain English and then as relational algebra. Your queries should include joins and at least one should include an aggregate function. At least one of your queries should use “extra” entities you added to your model in Checkpoint 01.

Queries on WAREHOUSE(From Checkpoint 01 extra entity)

- List all book titles in the W_Code is 100.

$$\pi_{(Title)}(\text{BOOK} \bowtie_{(BOOK.ISBN=STORE\_IN.ISBN)} ((\delta_{(W\_code=100)}(STORE\_IN))))$$

Queries on CATEGORY(From Checkpoint 01 extra entity)

- List all books on biography.

$$\pi_{(Title)}(\text{BOOK} \bowtie_{(BOOK.ISBN=CATEGORY.ISBN)} ((\delta_{(Genre =\text{”Biography”})} CATEGORY)))$$

Queries on ORDER

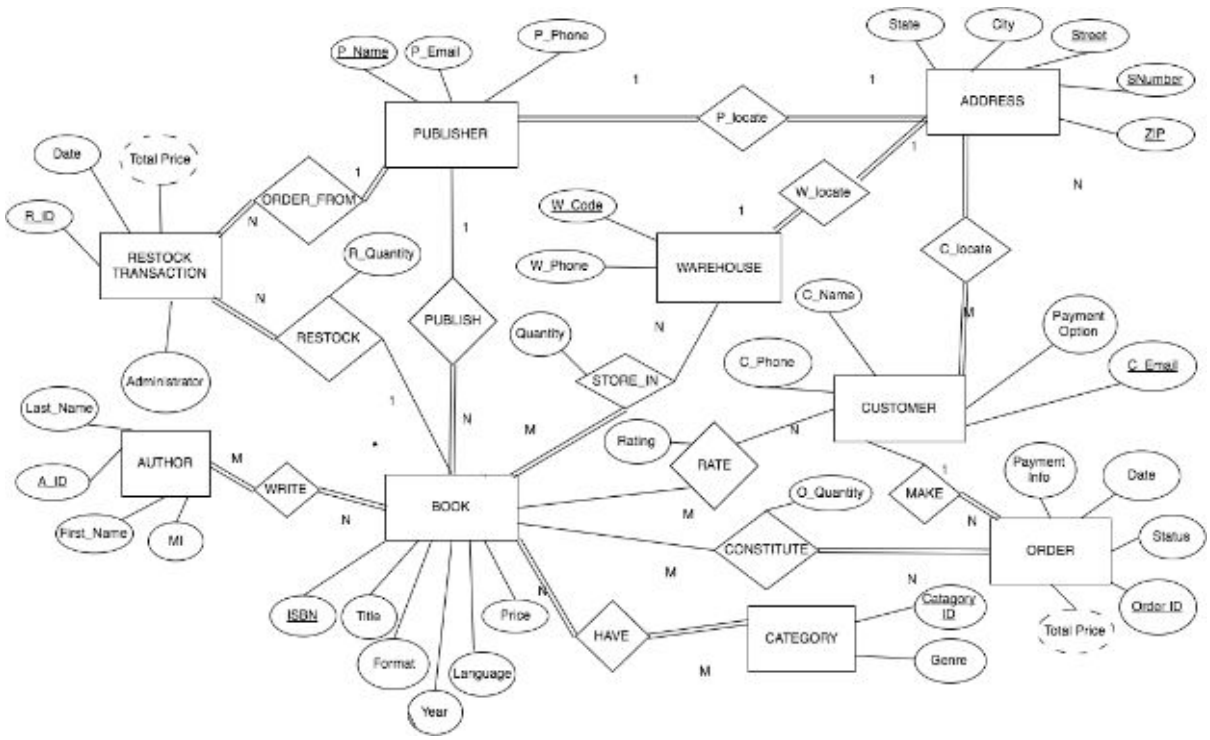
- Compute the number of order and average total price in ORDER entity.

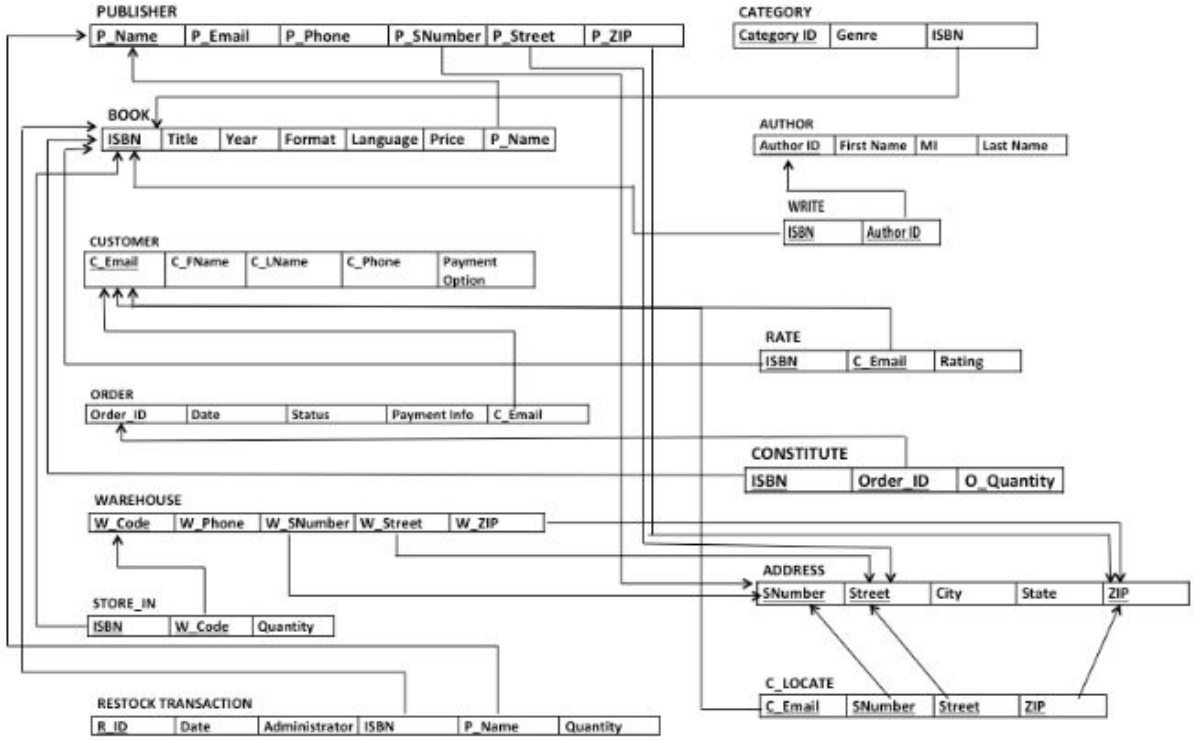
$$\mathcal{F}_{COUNT(Order\_ID),AVERAGE(Total\ Price)}(ORDER)$$

Name: Chen Cai, Tingwei Duan, Hui Li, En-Ju Lin

Date: Jun 15, 2016

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. **If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models**





2. Given your relational schema, create a text file containing the SQL code to create your database schema. Use this SQL to create a database in SQLite. Populate this database with the data provided for the project as well as 20 sample records for each table that does not contain data provided in the original project documents.

```
CREATE TABLE PUBLISHER

( P_Name     VARCHAR(30)  NOT NULL,

  P_Email    VARCHAR(30) ,

  P_Phone    CHAR,

  P_SNumber  INT          NOT NULL,

  P_Street   VARCHAR(30)  NOT NULL,

  P_ZIP      INT          NOT NULL,

  PRIMARY KEY(P_Name),

  FOREIGN KEY(P_SNumber) REFERENCES ADDRESS(SNumber),

  FOREIGN KEY(P_Street)  REFERENCES ADDRESS(Street),

  FOREIGN KEY(P_ZIP)     REFERENCES ADDRESS(ZIP)

);
```

```
CREATE TABLE BOOK
( ISBN       INT          NOT NULL,
  Title      VARCHAR(30),
  Year       INT,
  Format     VARCHAR(10),
  Language   VARCHAR(10),
  Price      INT,
  P_Name     VARCHAR(30)  NOT NULL,
  PRIMARY KEY(ISBN),
  FOREIGN KEY(P_Name) REFERENCES PUBLISHER(P_Name)
);


CREATE TABLE CUSTOMER
( C_Email        VARCHAR(30) NOT NULL,
  C_FName        VARCHAR(30),
  C_LName        VARCHAR(30),
  C_Phone        INT,
  Payment Option VARCHAR(30),
  PRIMARY KEY(C_Email)
);


CREATE TABLE ORDERA
( Order_ID       INT         NOT NULL,
  Date           CHAR,
  State          VARCHAR(10),
  Status         VARCHAR(10),
  Payment Info   VARCHAR(20),
  C_Email        VARCHAR(30) NOT NULL,
  FOREIGN KEY(C_Email) REFERENCES CUSTOMER(C_Email)
```

```
);


CREATE TABLE WAREHOUSE
( W_Code       INT         NOT NULL,
  W_Phone      CHAR(15),
  W_SNumber    INT         NOT NULL,
  W_Street     VARCHAR(30) NOT NULL,
  W_ZIP        INT         NOT NULL,
  FOREIGN KEY(W_SNumber) REFERENCES ADDRESS(SNumber),
  FOREIGN KEY(W_Street) REFERENCES ADDRESS(Street),
  FOREIGN KEY(W_ZIP) REFERENCES ADDRESS(ZIP)
);
CREATE TABLE STORE_IN
( ISBN         INT         NOT NULL,
  W_Code       INT         NOT NULL,
  Quantity     INT,
  FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN),
  FOREIGN KEY(W_Code) REFERENCES WAREHOUSE(W_Code)
);
CREATE TABLE RESTOCK_TRANSACTION
( R_ID         INT         NOT NULL,
  Date         CHAR(10),
  Administrator VARCHAR(30),
  ISBN         INT         NOT NULL,
  P_Name       VARCHAR(30)  NOT NULL,
  Quantity     INT,
  PRIMARY KEY(R_ID),
  FOREIGN KEY(ISBN)   REFERENCES BOOK(ISBN),
  FOREIGN KEY (P_NAME) REFERENCES PUBLISHER(P_Name)
```

```
);


CREATE TABLE CATEGORY
( Category_ID   INT       NOT NULL,
  Genre         VARCHAR(10),
  ISBN          INT         NOT NULL,
  PRIMARY KEY(Category_ID),
  FOREIGN KEY(ISBN) REFERENCES PUBLISHER(P_Name)
);


CREATE TABLE AUTHOR
( Author_ID     INT         NOT NULL,
  First_Name    VARCHAR(30),
  MI            VARCHAR(30),
  Last_Name     VARCHAR(30),
  PRIMARY KEY(Author_ID)
);


CREATE TABLE WRITE
( ISBN          INT         NOT NULL,
  Author_ID     INT         NOT NULL,
  FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN),
  FOREIGN KEY(Author_ID) REFERENCES AUTHOR(Author_ID)
);


CREATE TABLE RATE
( ISBN          INT         NOT NULL,
  C_Email       VARCHAR(30)  NOT NULL,
  Rating        INT,
```

```
 FOREIGN KEY(ISBN)   REFERENCES BOOK(ISBN),
 FOREIGN KEY(C_Email) REFERENCES CUSTOMER(C_Email)
);

CREATE TABLE CONSTITUTE
( ISBN          INT          NOT NULL,
 Order_ID       INT          NOT NULL ,
 O_Quantity     INT,
 FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN),
 FOREIGN KEY(Order_ID) REFERENCES ORDERA(Order_ID)
);

CREATE TABLE ADDRESS
( SNumber       INT          NOT NULL,
 Street         VARCHAR(30)  NOT NULL,
 City           VARCHAR(30),
 State          VARCHAR(30),
 ZIP            INT          NOT NULL,
 PRIMARY KEY(Street)
);

CREATE TABLE C_LOCATE
( C_Email       VARCHAR(30)  NOT NULL,
 SNumber        INT          NOT NULL,
 Street         VARCHAR(30)  NOT NULL,
 ZIP            INT          NOT NULL,
 FOREIGN KEY(C_Email) REFERENCES CUSTOMER(C_Email),
 FOREIGN KEY(SNumber) REFERENCES ADDRESS(SNumber),
 FOREIGN KEY(Street)  REFERENCES ADDRESS(Street),
```

```
 FOREIGN KEY(ZIP)     REFERENCES ADDRESS(ZIP)

);
```

3. Given your relational schema, provide the SQL to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named “WorksheetTwoSimpleQueries.txt”:

a. Find the titles of all books by Pratchett that cost less than $10

```
SELECT Title

FROM BOOK AS B, WRITE AS W, AUTHOR AS A

WHERE

                A.Last_Name = 'Pratchett'      AND

                B.Price < 10              AND

                B.ISBN = W.ISBN                AND

                W.Author_ID = A.Author_ID;
```

b. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

```
SELECT Title, Date

FROM CUSTOMER AS CU, BOOK AS B, ORDER AS O, CONSTITUTE AS C

WHERE

        UU.C_Email = ‘cai.4@osu.edu’Email = “cai.4@osu.edu”      AND

        UU.CC_Email = O.C_Email    AND

        O.Order_ID = C.Order_ID         AND

        C.ISBN = B.ISBN;
```

c. Find the titles and ISBNs for all books with less than 5 copies in stock

```
SELECT Title, ISBN

FROM BOOK AS B, STORE_IN AS S
```

```
WHERE
        B.ISBN = S.ISBN       AND
        S.Quantity < 5 ;
```

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

```
SELECT C_FName, C_LName, Title
FROM CUSTOMER AS UU, BOOK AS B, WRITE AS W, AUTHOR AS A, ORDER AS O, CONSTITUTE AS C
WHERE
    U.C_Email  = O.C_Email          AND
    O.Order_ID = C.Order_ID         AND
    C.ISBN = B.ISBN             AND
    A.Last_Name = 'Pratchett'       AND
    A.ID = W.ID         AND
    W.ISBN = B.ISBN;
```

e. Find the total number of books purchased by a single customer (you choose how to designate the customer)

```
SELECT sum(ORDERCOUNT)
FROM ORDERCOUNT AS Oc,
        (
          SELECT count(BOOK)
          FROM CUSTOMER AS Cu, ORDER AS O, CONSTITUTE AS C
          WHERE
                Cu.C_Email = O.C_Email      AND
                O.Order_ID = C.Order_ID
        ) AS ORDERCOUNT
```

```
WHERE
        Oc.C_Email = 'buck.1@osu.edu'
```

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

```
SELECT C_Email, sum(ORDERCOUNT)
FROM ORDERCOUNT AS Oc,
        (
          SELECT count(BOOK)
          FROM CUSTOMER AS Cu, ORDER AS O, CONSTITUTE AS C
          WHERE
                Cu.C_Email = O.C_Email      AND
                O.Order_ID = C.Order_ID
        ) AS ORDERCOUNT
WHERE MAX(ORDERCOUNT)
```

4. For Project Checkpoint 02, you were asked to come up with three additional interesting queries that your database can provide. Give what those queries are supposed to retrieve in plain English, as relational algebra and then as SQL. Your queries should include joins and at least one should include an aggregate function, and they should be the same as the queries you outlined for Worksheet 02. If you were instructed to fix the queries in Checkpoint 02, make sure you use the fixed queries here. These queries should be provided in a plain text file named "WorksheetTwoExtraQueries.txt".

Three additional queries from Checkpoint 02:

Queries on WAREHOUSE

- List all book titles in the W_Code is 100.
- Description: the query will retrieve all book titles of a particular warehouse with W_Code 100.
- Relational algebra:

$$\pi_{(Title)}(\text{BOOK} \bowtie_{(BOOK.ISBN=STORE\_IN.ISBN)} ((\delta_{(W\_code=100)}(STORE\_IN))))$$

- SQL:

```
SELECT    Title
FROM      STORE_IN,  BOOK
WHERE     W_Code = '100'
          AND STORE_IN.ISBN = BOOK.ISBN ;
```

Queries on CATEGORY

- List all books on biography.
- Relational algebra:

$\pi_{(Title)}$ (BOOK $\bowtie_{(BOOK.ISBN=CATEGORY.ISBN)}$ (($\delta_{(Genre =\text{"Biography"})}$ CATEGORY)))

- Description: the query will retrieve all book titles that belongs to the category 'biography'
- SQL:

```
SELECT    Title
FROM      CATEGORY AS C, BOOK AS B
WHERE     C.Genre = "Biography"        AND
          B.ISBN = C.ISBN
```

Queries on ORDER

- Description: Compute the number of orders and average total price (average sale per order)

- Relational algebra:

BOOKPRICE ← $\pi_{(Order\_ID, ISBN, Price, O\_Quantity)}$(CONSTITUTE $\bowtie_{(CONSTITUTE..ISBN=BOOK.ISBN)}$(BOOK))

$\rho_{TOTALPRICE\ (Order\_ID, Order\_Total\_Price)}$ ($_{Order\_ID}\mathcal{F}_{sum(Price * O\_Quantity)}$ (BOOKPRICE))

AVERAGE ← $\mathcal{F}_{COUNT(Order\_ID), AVERAGE\ (Order\_Total\_Price)}$(TOTALPRICE)

- SQL:

```
CREATE VIEW TOTAL_ORDER (OrderCount, TotalSale)
SELECT    count (distinct C.Order_ID),  sum (B.Price * C.O_Quantity)
FROM      CONSTITUTE AS C, BOOK AS B
WHERE     C.ISBN = B.ISBN

CREATE VIEW AVERAGE_ORDER
SELECT    TotalSale / OrderCount AS Average_Order
FROM      TOTAL_ORDER
```

5. Given your relational schema, provide the SQL for the following more advanced queries. These queries may require you to use techniques such as nesting, aggregation using having clauses, and other techniques . If your database schema does not contain the information to answer to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. **Note that if your database does contain the information but in non-aggregated form, you should NOT revise your model but instead figure out how to aggregate it for the query!** These queries should be provided in a plain text file named “WorksheetTwoAdvancedQueries.txt”.

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

```
SELECT    C_FName, C_LName, SUM(B.Price * C.O_Quantity)
FROM      BOOK AS B, CONSTITUTE AS C, CUSTOMER AS U,
ORDER AS O
WHERE     B.ISBN = C.ISBN AND O.C_Email = U.C_Email
          AND C.Order_ID = O.Order_ID
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       IND_SALE
SELECT    C_FName, C_LName, C_Email, sum(B.Price * C.O_Quantity)
AS Indivi_Sale
FROM      BOOK AS B, CONSTITUTE AS C, CUSTOMER AS U,
ORDER AS O
WHERE     B.ISBN = C.ISBN AND O.C_Email = U.C_Email
          AND C.Order_ID = O.Order_ID
```

```
GROUP BY    C_Email;

CREATE VIEW         AVG_SALE
SELECT      sum(Indivi_Sale) AS Total_Sale, count (distinct C.Email) AS
Count,  avg (Total_Sale / Count) AS Avg_Sale
FROM        IND_SALE


CREATE VIEW         MORE_AVG
SELECT      C_FName, C_LName, C_Email
FROM        IND_SALE
GROUP BY    C_Email
HAVING      Indivi_Sale > Avg_Sale ;
```

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.

```
SELECT      Title, COUNT(O_Quantity)
FROM        BOOK AS B, CONSTITUTE AS C
WHERE       B.ISBN = C.ISBN
GROUP BY    ISBN
ORDER BY    O_Quantity 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.

```
SELECT      Title, SUM(BOOK.Price * CONSTITUTE.O_Quantity) AS Total_Sale
FROM        BOOK AS B, CONSTITUTE AS C
WHERE       B.ISBN = C.ISBN
GROUP BY    ISBN
ORDER BY    Total_Sale DESC;
```

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

```
SELECT      Author_ID, First Name, MI, Last Name, COUNT(O_Quantity)
```

```
FROM        AUTHOR AS A, CONSTITUTE AS C, WRITE AS W
WHERE       W.ISBN = C.ISBN AND W.Author_ID = A.Author_ID.
GROUP BY    Author_ID
ORDER BY    O_Quantity DESC;
```

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 P_AUTH1
SELECT      Author_ID, First Name, MI, Last Name, MAX( SUM(BOOK.Price *
CONSTITUTE.O_Quantity))
 FROM       AUTHOR AS A, BOOK AS B, CONSTITUTE AS C, WRITE AS W
WHERE       B.ISBN = C.ISBN AND C.ISBN = W.ISBN
            AND W.Author_ID = A.Author_ID.
GROUP BY    Author_ID ;
```

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

```
CREATE VIEW P_AUTH_BOOK
SELECT      C_FName, C_LName, C_Email, C_Phone
FROM        P_AUTH1 AS P, CONSTITUTE AS C, WRITE AS W
            ORDER AS O, CUSTOMER AS U
WHERE       P.Author_ID = W.Author_ID
            AND W.ISBN = C.ISBN
            AND C.Order_Id = O.Order_ID
            AND O.C_Email = U.C_Email
GROUP BY    U.C_Email ;
```

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

```
CREATE VIEW         IND_SALE
SELECT      C_FName, C_LName, C_Email, sum(B.Price * C.O_Quantity)
AS Indivi_Sale
FROM        BOOK AS B, CONSTITUTE AS C, CUSTOMER AS U,
ORDER AS O
```

```
WHERE       B.ISBN = C.ISBN AND O.C_Email = U.C_Email
            AND C.Order_ID = O.Order_ID
GROUP BY    C_Email;


CREATE VIEW     MORE_AVG
SELECT      C_FName, C_LName, C_Email
FROM        IND_SALE
GROUP BY    C_Email
HAVING      Indivi_Sale > Avg_Sale ;

CREATE VIEW     AUTHOR_LIST
SELECT          Author_ID, First_Name, Last_Name
FROM            AUTHOR AS A, WRITE AS W, CONSTITUTE AS C,
ORDER AS O, MORE_AVG AS M
WHERE           M.C_Email = O.C_Email
                AND O.Order_ID = C.Order_ID
                AND C.ISBN = W.ISBN
                AND W.Author_ID = A.Author ID
GROUP BY        A.Author_ID
```

CSE 3241 Project Checkpoint 03_Individual Feedback

1. ER & Schema

Too many relationships are total participation though (make some partial). (See Checkpoint 03_Revised for the new relational model diagram and schema)

2. DB Creation File

Include a separate .txt file with the SQL used to create the tables. Maybe you should rename ORDER to C_ORDER or something like that to reduce issues (since ORDER is keyword in SQL).

3. Simple Queries

Your .txt files were hard to read/use. Make sure they are formatted neatly. Also, add more sample records to that the queries don't return empty tables. Make sure that references to ORDER match the name (ORDERA, or C_ORDER is even better).

3b seems to contain a syntax error, but I can't find it.

In 3c, ISBN is ambiguous. Use B.ISBN.

In 3e, you should sum CONSTITUTE.o_quantity for all items in all a customer's orders.

Similar issue with 3f.

4. Extra Queries

Make .txt file neater. Fix syntax issues. Looks good otherwise.

5. Advanced Queries

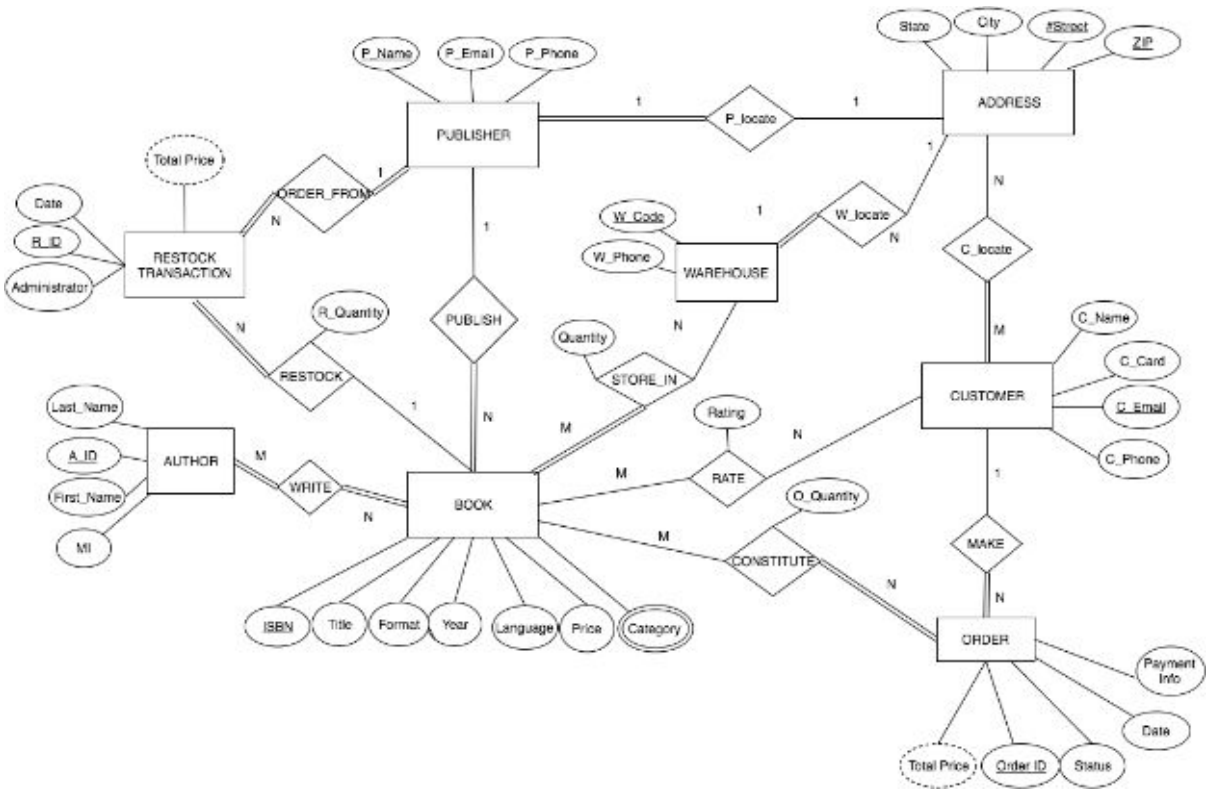
Make .txt file neater. Fix syntax issues.
In 5f, you should split up doing the sum and finding the max sum into two parts.
BINARY SQLITE OF DB
Do not include the table 'checkpoint3'. Add more sample records.
(See Final SQL Database for the new SQL)

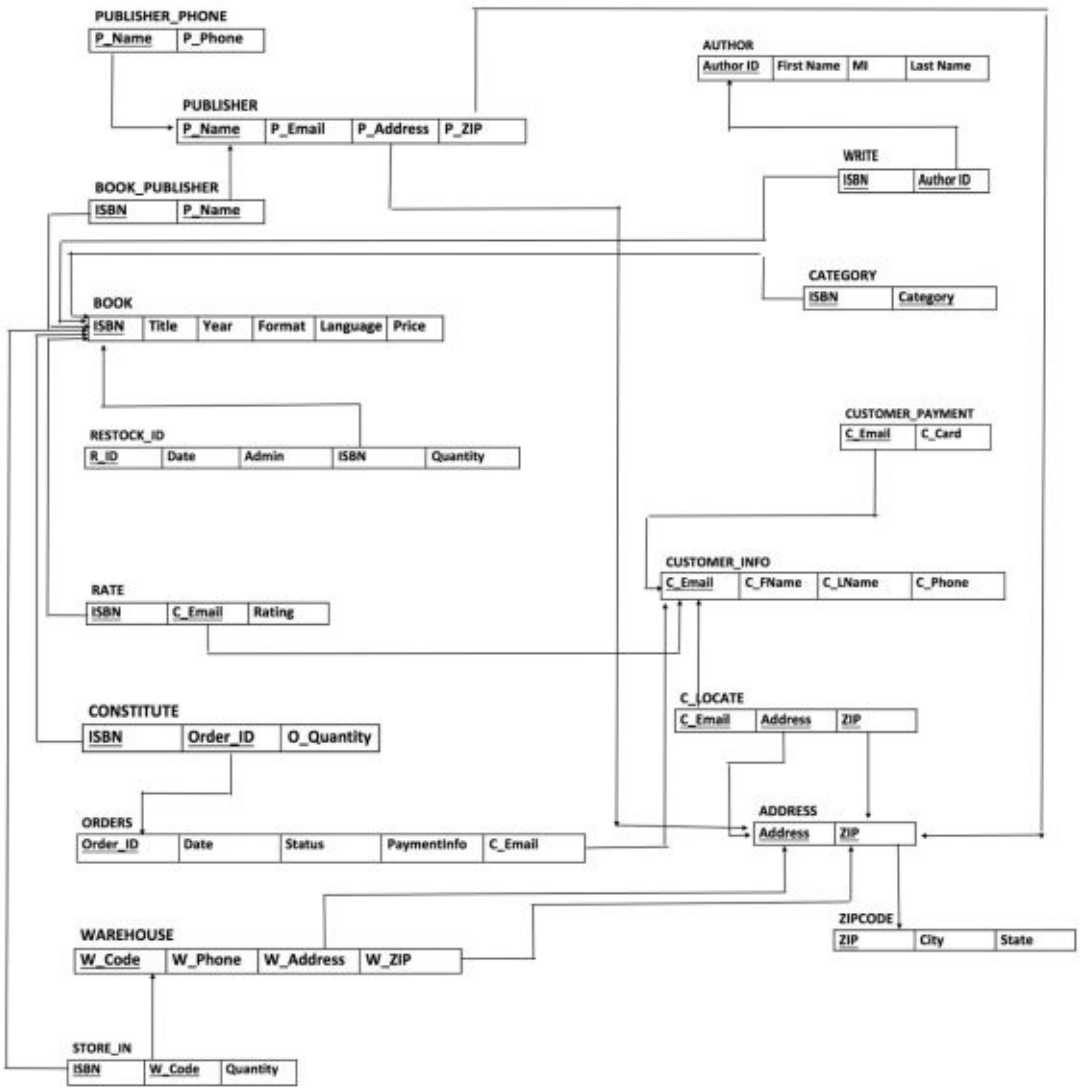
CSE 3241 Project Checkpoint 03 – Revised

Name: Chen Cai, Tingwei Duan, Hui Li, En-Ju Lin

Date: Jun 15, 2016

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 02. **If you were instructed to change the model for Project Checkpoint 02, make sure you use the revised versions of your models**





2. Given your relational schema, create a text file containing the SQL code to create your database schema. Use this SQL to create a database in SQLite. Populate this database with the data provided for the project as well as 20 sample records for each table that does not contain data provided in the original project documents.

```
CREATE TABLE PUBLISHER

( P_Name     VARCHAR(30)  NOT NULL,

  P_Email    VARCHAR(30) ,

  P_Phone    CHAR,
```

```
 P_SNumber  INT          NOT NULL,
 P_Street   VARCHAR(30)  NOT NULL,
 P_ZIP      INT          NOT NULL,
 PRIMARY KEY(P_Name),
 FOREIGN KEY(P_SNumber) REFERENCES ADDRESS(SNumber),
 FOREIGN KEY(P_Street)  REFERENCES ADDRESS(Street),
 FOREIGN KEY(P_ZIP)     REFERENCES ADDRESS(ZIP)
);

CREATE TABLE BOOK
( ISBN       INT          NOT NULL,
 Title      VARCHAR(30),
 Year       INT,
 Format     VARCHAR(10),
 Language   VARCHAR(10),
 Price      INT,
 P_Name     VARCHAR(30)  NOT NULL,
 PRIMARY KEY(ISBN),
 FOREIGN KEY(P_Name) REFERENCES PUBLISHER(P_Name)
);

CREATE TABLE CUSTOMER
( C_Email        VARCHAR(30) NOT NULL,
 C_FName        VARCHAR(30),
 C_LName        VARCHAR(30),
 C_Phone        INT,
 Payment Option VARCHAR(30),
 PRIMARY KEY(C_Email)
);
```

```
CREATE TABLE ORDERA
( Order_ID        INT         NOT NULL,
  Date            CHAR,
  State           VARCHAR(10),
  Status          VARCHAR(10),
  Payment Info    VARCHAR(20),
  C_Email         VARCHAR(30) NOT NULL,
  FOREIGN KEY(C_Email) REFERENCES CUSTOMER(C_Email)
);


CREATE TABLE WAREHOUSE
( W_Code       INT         NOT NULL,
  W_Phone      CHAR(15),
  W_SNumber    INT         NOT NULL,
  W_Street     VARCHAR(30) NOT NULL,
  W_ZIP        INT         NOT NULL,
  FOREIGN KEY(W_SNumber) REFERENCES ADDRESS(SNumber),
  FOREIGN KEY(W_Street) REFERENCES ADDRESS(Street),
  FOREIGN KEY(W_ZIP) REFERENCES ADDRESS(ZIP)
);
CREATE TABLE STORE_IN
( ISBN         INT         NOT NULL,
  W_Code       INT         NOT NULL,
  Quantity     INT,
  FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN),
  FOREIGN KEY(W_Code) REFERENCES WAREHOUSE(W_Code)
);
CREATE TABLE RESTOCK_TRANSACTION
( R_ID         INT         NOT NULL,
```

```
 Date         CHAR(10),
 Administrator VARCHAR(30),
 ISBN         INT          NOT NULL,
 P_Name       VARCHAR(30)  NOT NULL,
 Quantity     INT,
 PRIMARY KEY(R_ID),
 FOREIGN KEY(ISBN)   REFERENCES BOOK(ISBN),
 FOREIGN KEY (P_NAME) REFERENCES PUBLISHER(P_Name)
);

CREATE TABLE CATEGORY
( Category_ID   INT       NOT NULL,
 Genre         VARCHAR(10),
 ISBN          INT         NOT NULL,
 PRIMARY KEY(Category_ID),
 FOREIGN KEY(ISBN) REFERENCES PUBLISHER(P_Name)
);

CREATE TABLE AUTHOR
( Author_ID     INT          NOT NULL,
 First_Name    VARCHAR(30),
 MI            VARCHAR(30),
 Last_Name     VARCHAR(30),
 PRIMARY KEY(Author_ID)
);

CREATE TABLE WRITE
( ISBN          INT          NOT NULL,
 Author_ID     INT          NOT NULL,
```

```
 FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN),
 FOREIGN KEY(Author_ID) REFERENCES AUTHOR(Author_ID)
);

CREATE TABLE RATE
( ISBN          INT          NOT NULL,
 C_Email       VARCHAR(30)  NOT NULL,
 Rating        INT,
 FOREIGN KEY(ISBN)   REFERENCES BOOK(ISBN),
 FOREIGN KEY(C_Email) REFERENCES CUSTOMER(C_Email)
);

CREATE TABLE CONSTITUTE
( ISBN          INT          NOT NULL,
 Order_ID        INT          NOT NULL ,
 O_Quantity    INT,
 FOREIGN KEY(ISBN) REFERENCES BOOK(ISBN),
 FOREIGN KEY(Order_ID) REFERENCES ORDERA(Order_ID)
);

CREATE TABLE ADDRESS
( SNumber       INT          NOT NULL,
 Street        VARCHAR(30)  NOT NULL,
 City          VARCHAR(30),
 State         VARCHAR(30),
 ZIP           INT          NOT NULL,
 PRIMARY KEY(Street)
);
```

```
CREATE TABLE C_LOCATE
( C_Email      VARCHAR(30)  NOT NULL,
 SNumber      INT          NOT NULL,
 Street       VARCHAR(30)  NOT NULL,
 ZIP          INT          NOT NULL,
 FOREIGN KEY(C_Email) REFERENCES CUSTOMER(C_Email),
 FOREIGN KEY(SNumber) REFERENCES ADDRESS(SNumber),
 FOREIGN KEY(Street)  REFERENCES ADDRESS(Street),
 FOREIGN KEY(ZIP)     REFERENCES ADDRESS(ZIP)
);
```

3. Given your relational schema, provide the SQL to perform the following queries. If your schema cannot provide answers to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. These queries should be provided in a plain text file named “WorksheetTwoSimpleQueries.txt”:

a. Find the titles of all books by Pratchett that cost less than $10

```
SELECT Title
FROM BOOK AS B, WRITE AS W, AUTHOR AS A
WHERE
                A.Last_Name = 'Pratchett'     AND
                B.Price < 10              AND
                B.ISBN = W.ISBN               AND
                W.Author_ID = A.Author_ID;
```

b. Give all the titles and their dates of purchase made by a single customer (you choose how to designate the customer)

```
SELECT Title, Date
FROM CUSTOMER AS CU, BOOK AS B, ORDER AS O, CONSTITUTE AS C
WHERE
```

```
        UU.C_Email = 'cai.4@osu.edu'Email = "cai.4@osu.edu"      AND

        UU.CC_Email = O.C_Email    AND

        O.Order_ID = C.Order_ID         AND

        C.ISBN = B.ISBN;
```

c. Find the titles and ISBNs for all books with less than 5 copies in stock

```
SELECT Title, ISBN

FROM BOOK AS B, STORE_IN AS S

WHERE

        B.ISBN = S.ISBN        AND

        S.Quantity < 5 ;
```

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

```
SELECT C_FName, C_LName, Title

FROM CUSTOMER AS UU, BOOK AS B, WRITE AS W, AUTHOR AS A, ORDER AS O,
CONSTITUTE AS C

WHERE

    U.C_Email  = O.C_Email             AND

    O.Order_ID = C.Order_ID            AND

    C.ISBN = B.ISBN              AND

    A.Last_Name = 'Pratchett'          AND

    A.ID = W.ID          AND

    W.ISBN = B.ISBN;
```

e. Find the total number of books purchased by a single customer (you choose how to designate the customer)

```
SELECT sum(ORDERCOUNT)

FROM ORDERCOUNT AS Oc,
```

```
        (
          SELECT count(BOOK)
          FROM CUSTOMER AS Cu, ORDER AS O, CONSTITUTE AS C
          WHERE
                Cu.C_Email = O.C_Email      AND
                O.Order_ID = C.Order_ID
        ) AS ORDERCOUNT
WHERE
        Oc.C_Email = 'buck.1@osu.edu'
```

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

```
SELECT C_Email, sum(ORDERCOUNT)
FROM ORDERCOUNT AS Oc,
        (
          SELECT count(BOOK)
          FROM CUSTOMER AS Cu, ORDER AS O, CONSTITUTE AS C
          WHERE
                Cu.C_Email = O.C_Email      AND
                O.Order_ID = C.Order_ID
        ) AS ORDERCOUNT
WHERE MAX(ORDERCOUNT)
```

4. For Project Checkpoint 02, you were asked to come up with three additional interesting queries that your database can provide. Give what those queries are supposed to retrieve in plain English, as relational algebra and then as SQL. Your queries should include joins and at least one should include an aggregate function, and they should be the same as the queries you outlined for Worksheet 02. If you were instructed to fix the queries in

Checkpoint 02, make sure you use the fixed queries here. These queries should be provided in a plain text file named “WorksheetTwoExtraQueries.txt”.

Three additional queries from Checkpoint 02:

Queries on WAREHOUSE

- List all book titles in the W_Code is 100.
- Description: the query will retrieve all book titles of a particular warehouse with W_Code 100.
- Relational algebra:

$$\pi_{(Title)}(BOOK \bowtie_{(BOOK.ISBN=STORE\_IN.ISBN)} ((\delta_{(W\_code=100)}(STORE\_IN))))$$

- SQL:

```
SELECT    Title
FROM      STORE_IN, BOOK
WHERE     W_Code ='W_100'
          AND STORE_IN.ISBN = BOOK.ISBN ;
```

Queries on CATEGORY

- List all books on biography.
- Relational algebra:

$$\pi(Title)(BOOK \bowtie(BOOK.ISBN=CATEGORY.ISBN) ((\delta(Category =”Biography”) CATEGORY)))$$

- Description: the query will retrieve all book titles that belongs to the category ‘biography’
- SQL:

```
SELECT    Title
FROM      CATEGORY AS C, BOOK AS B
WHERE     C.Category = “Biography”    AND
          B.ISBN = C.ISBN
```

Queries on ORDER

- Description: Compute the number of orders and average total price (average sale per order)

- Relational algebra:

BOOKPRICE ← π(Order_ID, ISBN, Price, O_Quantity)(CONSTITUTE ⋈(CONSTITUTE..ISBN=BOOK.ISBN)(BOOK))

ρTOTALPRICE (Order_ID, Order_Total_Price) (Order_ID ℱsum(Price * O_Quantity) (BOOKPRICE))

AVERAGE ← ℱCOUNT(Order_ID), AVERAGE (Order_Total_Price)(TOTALPRICE)

- SQL:

```
CREATE VIEW TOTAL_ORDER (OrderCount, TotalSale)
SELECT    count (distinct C.Order_ID),  sum (B.Price * C.O_Quantity)
FROM      CONSTITUTE AS C, BOOK AS B
WHERE     C.ISBN = B.ISBN

CREATE VIEW AVERAGE_ORDER
SELECT    TotalSale / OrderCount AS Average_Order
FROM      TOTAL_ORDER
```

5. Given your relational schema, provide the SQL for the following more advanced queries. These queries may require you to use techniques such as nesting, aggregation using having clauses, and other techniques . If your database schema does not contain the information to answer to these queries, revise your ER Model and your relational schema to contain the appropriate information for these queries. **Note that if your database does contain the information but in non-aggregated form, you should NOT revise your model but instead figure out how to aggregate it for the query!** These queries should be provided in a plain text file named “WorksheetTwoAdvancedQueries.txt”.

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

```
SELECT    C_FName, C_LName, SUM(B.Price * C.O_Quantity)
FROM      BOOK AS B, CONSTITUTE AS C, CUSTOMER AS U,
ORDER AS O
WHERE     B.ISBN = C.ISBN AND O.C_Email = U.C_Email
          AND C.Order_ID = O.Order_ID
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     IND_SALE
SELECT      C_FName, C_LName, C_Email, sum(B.Price * C.O_Quantity)
AS Indivi_Sale
FROM        BOOK AS B, CONSTITUTE AS C, CUSTOMER AS U,
ORDER AS O
WHERE       B.ISBN = C.ISBN AND O.C_Email = U.C_Email
            AND C.Order_ID = O.Order_ID
GROUP BY    C_Email;

CREATE VIEW     AVG_SALE
SELECT      sum(Indivi_Sale) AS Total_Sale, count (distinct C.Email) AS
Count,  avg (Total_Sale / Count) AS Avg_Sale
FROM        IND_SALE

CREATE VIEW     MORE_AVG
SELECT      C_FName, C_LName, C_Email
FROM        IND_SALE
GROUP BY    C_Email
HAVING      Indivi_Sale > Avg_Sale ;
```

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.

```
SELECT      Title, COUNT(O_Quantity)
FROM        BOOK AS B, CONSTITUTE AS C
WHERE       B.ISBN = C.ISBN
GROUP BY    ISBN
ORDER BY    O_Quantity 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.

```
SELECT      Title, SUM(BOOK.Price * CONSTITUTE.O_Quantity) AS Total_Sale
FROM        BOOK AS B, CONSTITUTE AS C
WHERE       B.ISBN = C.ISBN
GROUP BY    ISBN
ORDER BY    Total_Sale DESC;
```

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

```
SELECT      Author_ID, First Name, MI, Last Name, COUNT(O_Quantity)
FROM        AUTHOR AS A, CONSTITUTE AS C, WRITE AS W
WHERE       W.ISBN = C.ISBN AND W.Author_ID = A.Author_ID.
GROUP BY    Author_ID
ORDER BY    O_Quantity DESC;
```

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 P_AUTH1
SELECT      Author_ID, First Name, MI, Last Name, MAX( SUM(BOOK.Price *
CONSTITUTE.O_Quantity))
 FROM       AUTHOR AS A, BOOK AS B, CONSTITUTE AS C, WRITE AS W
WHERE       B.ISBN = C.ISBN AND C.ISBN = W.ISBN
            AND W.Author_ID = A.Author_ID.
GROUP BY    Author_ID ;
```

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

```
CREATE VIEW P_AUTH_BOOK
SELECT      C_FName, C_LName, C_Email, C_Phone
FROM        P_AUTH1 AS P, CONSTITUTE AS C, WRITE AS W
            ORDER AS O, CUSTOMER AS U
WHERE       P.Author_ID = W.Author_ID
```

```
                    AND W.ISBN = C.ISBN
                    AND C.Order_Id = O.Order_ID
                    AND O.C_Email = U.C_Email
GROUP BY    U.C_Email ;
```

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

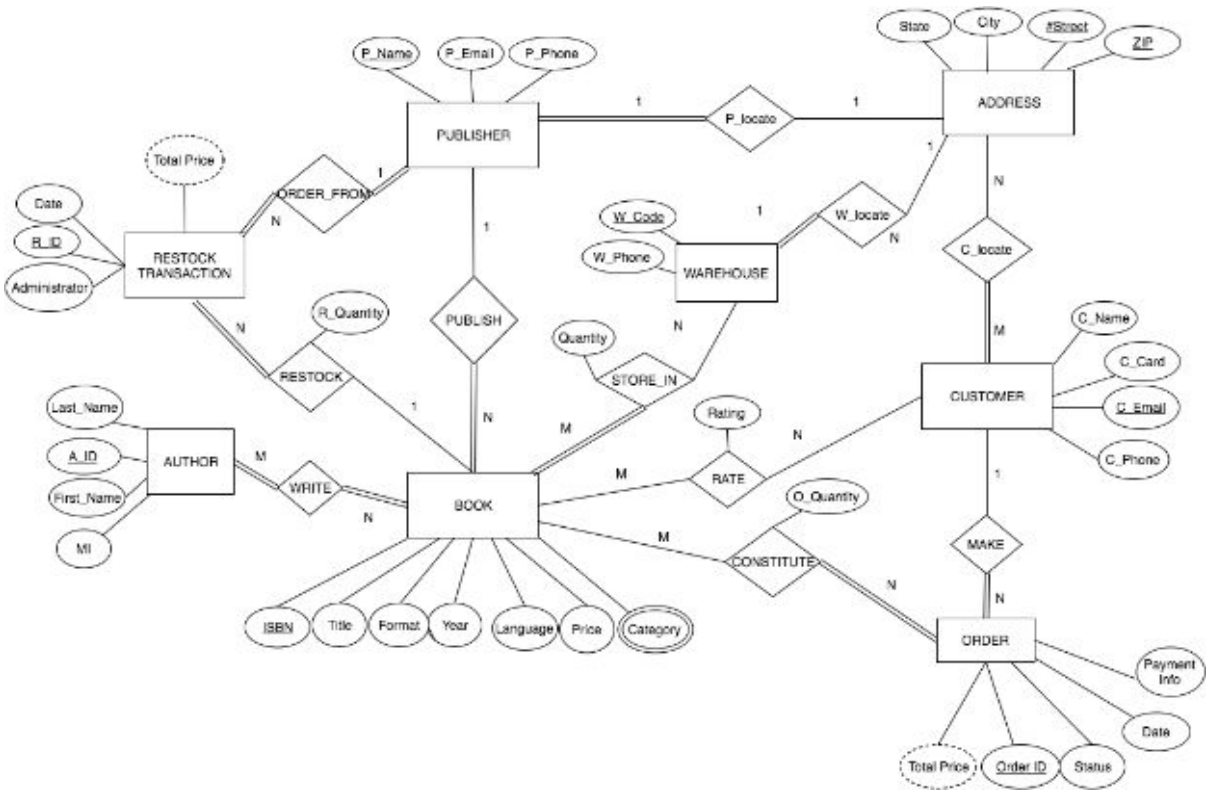
```
CREATE VIEW     IND_SALE
SELECT      C_FName, C_LName, C_Email, sum(B.Price * C.O_Quantity)
AS Indivi_Sale
FROM        BOOK AS B, CONSTITUTE AS C, CUSTOMER AS U,
ORDER AS O
WHERE       B.ISBN = C.ISBN AND O.C_Email = U.C_Email
            AND C.Order_ID = O.Order_ID
GROUP BY    C_Email;


CREATE VIEW     MORE_AVG
SELECT      C_FName, C_LName, C_Email
FROM        IND_SALE
GROUP BY    C_Email
HAVING      Indivi_Sale > Avg_Sale ;

CREATE VIEW     AUTHOR_LIST
SELECT          Author_ID, First_Name, Last_Name
FROM            AUTHOR AS A, WRITE AS W, CONSTITUTE AS C,
ORDER AS O, MORE_AVG AS M
WHERE           M.C_Email = O.C_Email
                AND O.Order_ID = C.Order_ID
                AND C.ISBN = W.ISBN
                AND W.Author_ID = A.Author ID
GROUP BY        A.Author_ID
```

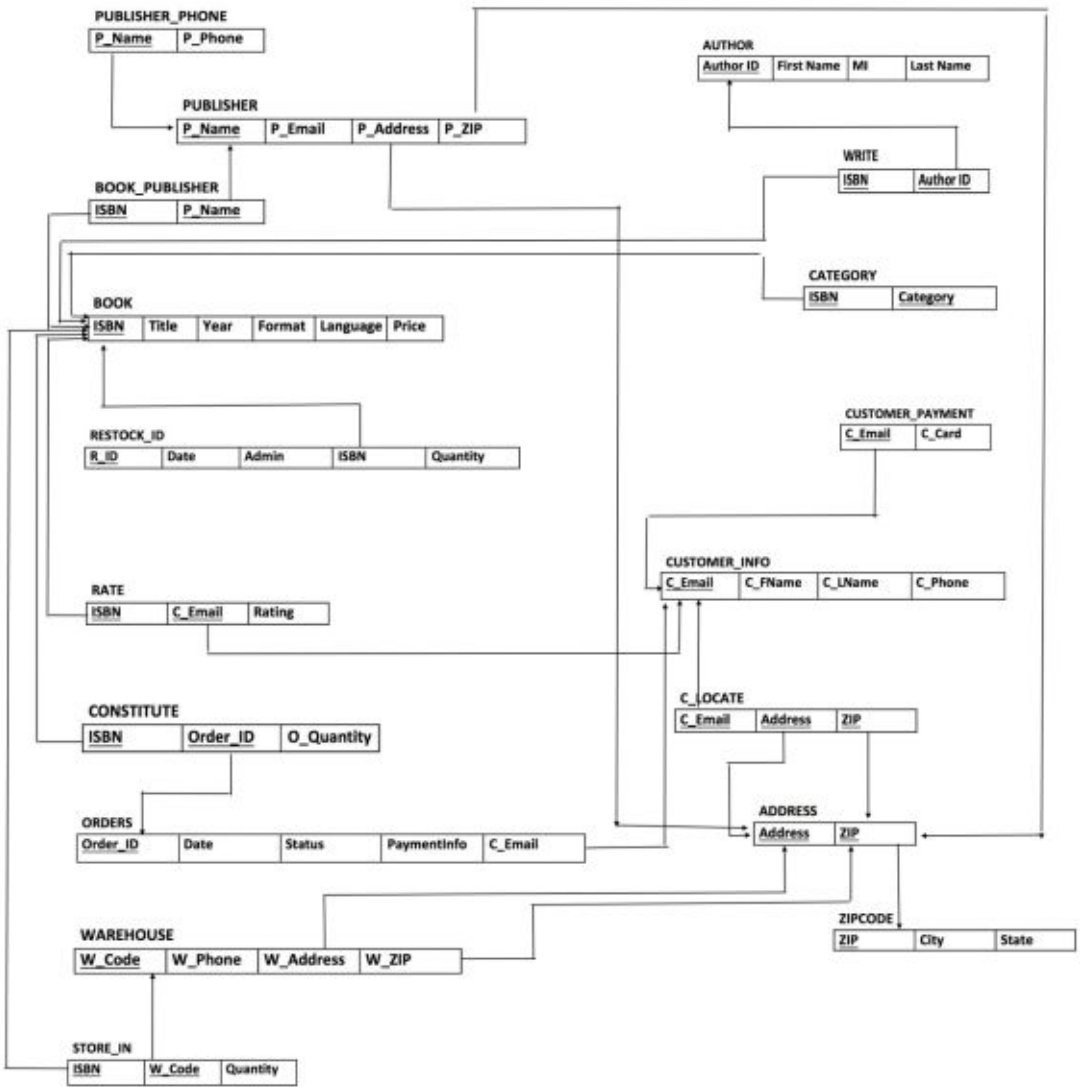
CSE 3241 Project Checkpoint 04 – Functional Dependencies and Normal Forms

Names Chen Cai, En-Ju Lin, Hui Li, Tingwei Duan

Date 6/24/2016

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 03. **If you were instructed to change the model for Project Checkpoint 03, make sure you use the revised versions of your models.**





2. For each relation schema in your model, indicate the functional dependencies. Think carefully about what you are modeling here - make sure you consider all the possible dependencies in each relation and not just the ones from your primary keys. For example, a customer's credit card number is unique, and so will uniquely identify a customer even if you have another key in the same table (in fact, if the customer can have multiple credit card numbers, the dependencies can get even more involved).

PUBLISHER:

{P_Name} → {P_Email, P_Phone, P_Address, P_ZIP}
{P_Email} → {P_Name, P_Phone, P_Address, P_ZIP}
{P_Address, P_ZIP} → {P_Name, P_Email, P_Phone}

Since P_Phone can be null, it cannot uniquely identify other attributes.

BOOK:

{ISBN} → {Title, Year, Format, Language, Price, P_Name}

CUSTOMER:

{C_Email} → {C_FName, C_LName, C_Phone, C_Card}

Since customer does not have to store a credit card (i.e. it can be null), it cannot always be used to uniquely identify other attributes, therefore functional dependency does not apply.

ORDER:

{Order_ID} → {Date, Status, PaymentInfo, C_Email}

PaymentInfo is for storing payment information such as credit card used. Since the same credit card can be used by multiple customers (e.g. same family), it cannot be used to uniquely identify the other attributes.

WAREHOUSE:

{W_Code} → {W_Phone, W_Address, W_ZIP}

STORE_IN

{ISBN, W_Code} → {Quantity}

RESTOCK TRANSACTION

{R_ID} → {Date, Administrator, ISBN, P_Name, Quantity}
{ISBN} → {P_Name}

CATEGORY

This is compound key so no functional dependency

AUTHOR:

{Author_ID} → {First Name, MI, Last Name}

WRITE:

This is compound key so no functional dependency

RATE:

{ISBN, C_Email} → {Rating}

CONSTITUTE:

{ISBN,ORDER_ID} → {O_Quantity}

ADDRESS: The ADDRESS relation is in 1NF because city and state are partially dependent on ZIP therefore it does not meet 2NF.

{Address, ZIP} → {City, State}

{ZIP} → {City, State}

C_LOCATE:
{C_Email} → {Address, ZIP}
{Address, ZIP} → {C_Email}

3. For each relation schema in your model, determine the highest normal form of the relation. If the relation is not in 3NF, rewrite your relation schema so that it is in at least 3NF.

PUBLISHER: Given that there could be multiple value entered for phone number, this relation could violate the condition of 1NF. To normalize to 3NF, we need to decompose it to two relations.

PUBLISHER
{P_Name, P_Email,P_Address, P_ZIP }

PUBLISHER_PHONE
{P_Name, P_Phone}

And actually it is now in BCNF.

BOOK: In 3NF, as every non-key attribute is non-fully and non-transitively dependent on the key.

BOOK
{ISBN, Title, Year, Format, Language, Price, P_Name}

Although no change is needed to achieve 3NF, due to the addition of BOOK_PUBLISHER relation below, we will take out the P_Name here to avoid redundancy.

BOOK
{ISBN, Title, Year, Format, Language, Price}

BOOK_PUBLISHER
{ISBN, P_Name}

CUSTOMER: The relation is in 1NF. For the database requirement, a customer can only have 1 phone number entered but can choose to save multiple or no credit card (C_Card) information. Therefore, we need to move the C_Card to a new relation to achieve 1NF and then the new schema would actually fulfill BCNF.

CUSTOMER_INFO

{C_Email, C_FName, C_LName, C_Phone}

CUSTOMER_PAYMENT
{C_Email, C_Card}

ADDRESS: The ADDRESS relation is in 1NF because city and state are partially dependent on ZIP therefore it does not meet 2NF. After modification below it is in BCNF.

ADDRESS
{Address, ZIP}

ZIPCODE
{ZIP, City, State}

ORDERS: This is in 3NF, since every non-key attribute is non-fully and non-transitively dependent on the key.

{Order_ID, Date, Status, PaymentInfo, C_Email}

WAREHOUSE: This is in 3NF, since every non-key attribute is non-fully and non-transitively dependent on the key.

{W_Code, W_Phone, W_Address, W_ZIP}

STORE_IN: This is in 3NF, since every non-key attribute is non-fully and non-transitively dependent on the key.

{ISBN, W_Code, Quantity}

RESTOCK TRANSACTION: This is in 2NF. Since P_Name is transitively dependent on the primary key R_ID through P_Name. To achieve 3NF, we will decompose this to two relations

RESTOCK_ID
{R_ID, Date, Admin, ISBN, Quantity}

BOOK_PUBLISHER
{ISBN, P_Name}

CATEGORY: BCNF (could be multivalued, but this is an all-key relation, which has no functional dependency)

CATEGORY
{ISBN, Category}

AUTHOR: BCNF

AUTHOR
{Author_ID, First Name, MI, Last Name}

WRITE: BCNF

WRITE
{ISBN, Author ID}

RATE: BCNF

RATE
{ISBN, C_Email, Rating}

CONSTITUTE: BCNF

CONSTITUTE
{ISBN, ORDER_ID, O_Quantity}

ADDRESS: The ADDRESS relation is in 1NF because city and state are partially dependent on ZIP therefore it does not meet 2NF.

ADDRESS
{Address, ZIP}

ZIP
{ZIP, City, State}

C_LOCATE: 3NF

{C_Email, Address, ZIP}

4. For each relation schema in your model that is in 3NF but not in BCNF, either rewrite the relation schema to BCNF or provide a short justification for why this relation should be an exception to the rule of putting relations into BCNF.

All relations after alternations are in BCNF.

5. For your database, propose at least two interesting views that can be built from your relations. These views must involve joining at least two tables together each and must include some kind of aggregation in the view. Each view must also be able to be described by a one or two sentence description in plain English. Provide the code for constructing your views along with the English language description of what the view is supposed to be providing.

1. (From Checkpoint 3 3.c)
   Find the titles and ISBNs for all books with less than 5 copies in stock

```
SELECT Title, B.ISBN

FROM BOOK AS B, STORE_IN AS S

WHERE

        B.ISBN = S.ISBN       AND

        S.Quantity < 5 ;
```

2. (From Checkpoint 3 3.e)

   Find the total number of books purchased by a single customer (you choose how to designate the customer)

```
SELECT sum(ORDERCOUNT)

FROM ORDERCOUNT AS Oc,

(

  SELECT sum(CONSTITUTE.O_Quantity)

  FROM CUSTOMER_INFO AS Cu, ORDERS AS O, CONSTITUTE AS C

  WHERE

        Cu.C_Email = O.C_Email      AND

        O.Order_ID = C.Order_ID

) AS ORDERCOUNT

WHERE

        Oc.C_Email = 'buck.1@osu.edu'
```

3. (From Checkpoint 3 5.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 P_MAX
SELECT        Author_ID, SUM(BOOK.Price * CONSTITUTE.O_Quantity) AS
Sum_P
 FROM         BOOK AS B, CONSTITUTE AS C, WRITE AS W
```

```
WHERE       B.ISBN = C.ISBN AND C.ISBN = W.ISBN
GROUP BY    Author_ID ;

SELECT      First Name, MI, Last Name, MAX(Sum_P)
 FROM       AUTHOR AS A, P_MAX AS P
WHERE       P.Author_ID = A.Author_ID;
```

CSE 3241 Project Checkpoint 04_Individual Feedback

(Cited from hardcopy returned on class)

Q1: (See Checkpoint 04_Revised for the new relational model diagram and shema)

· Try to make Category as an entity

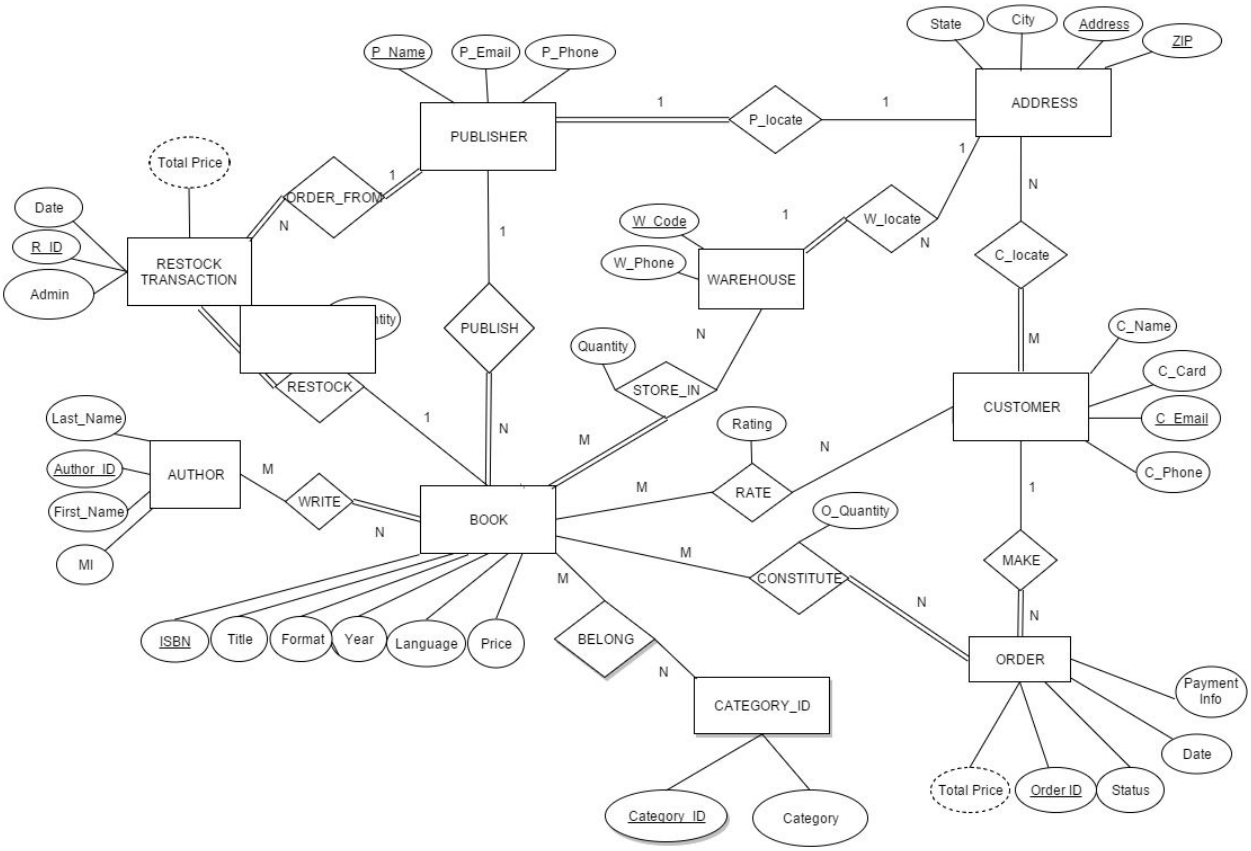
· Partial participation is better for Author writes book relationship

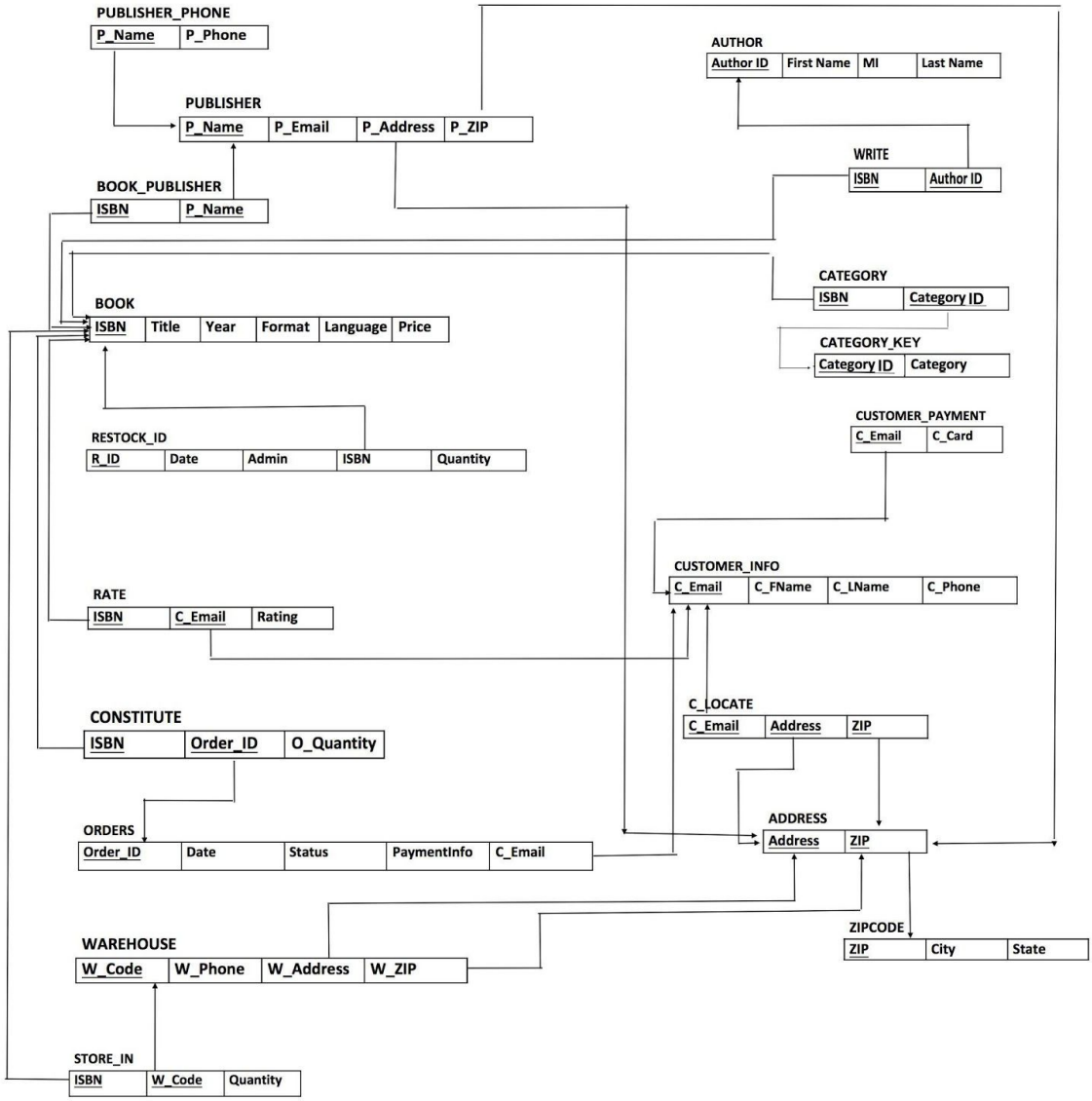
CSE 3241 Project Checkpoint 04 – Revised

Names Chen Cai, En-Ju Lin, Hui Li, Tingwei Duan

Date 6/24/2016

1. Provide a current version of your ER Diagram and Relational Model as per Project Checkpoint 03. **If you were instructed to change the model for Project Checkpoint 03, make sure you use the revised versions of your models.**


P_Name
P_Email
P_Phone
PUBLISHER
P_locate
State
City
Address
ZIP
ADDRESS
Total Price
ORDER_FROM
Date
R_ID
Admin
RESTOCK TRANSACTION
RESTOCK
PUBLISH
W_Code
W_Phone
WAREHOUSE
W_locate
C_locate
Quantity
STORE_IN
C_Name
C_Card
C_Email
C_Phone
CUSTOMER
Last_Name
Author_ID
First_Name
MI
AUTHOR
WRITE
BOOK
Rating
RATE
O_Quantity
CONSTITUTE
MAKE
ISBN
Title
Format
Year
Language
Price
BELONG
CATEGORY_ID
Category_ID
Category
ORDER
Payment Info
Date
Total Price
Order ID
Status



2. For each relation schema in your model, indicate the functional dependencies. Think carefully about what you are modeling here - make sure you consider all the possible dependencies in each relation and not just the ones from your primary keys. For example, a customer's credit card number is unique, and so will uniquely identify a customer even if you have another key in the same table (in fact, if the customer can have multiple credit card numbers, the dependencies can get even more involved).

PUBLISHER:

{P_Name} → {P_Email, P_Phone, P_Address, P_ZIP}
{P_Email} → {P_Name, P_Phone, P_Address, P_ZIP}
{P_Address, P_ZIP} → {P_Name, P_Email, P_Phone}

Since P_Phone can be null, it cannot uniquely identify other attributes.

BOOK:

{ISBN} → {Title, Year, Format, Language, Price, P_Name}

CUSTOMER:

{C_Email} → {C_FName, C_LName, C_Phone, C_Card}

Since customer does not have to store a credit card (i.e. it can be null), it cannot always be used to uniquely identify other attributes, therefore functional dependency does not apply.

ORDER:

{Order_ID} → {Date, Status, PaymentInfo, C_Email}

PaymentInfo is for storing payment information such as credit card used. Since the same credit card can be used by multiple customers (e.g. same family), it cannot be used to uniquely identify the other attributes.

WAREHOUSE:

{W_Code} → {W_Phone, W_Address, W_ZIP}

STORE_IN

{ISBN, W_Code} → {Quantity}

RESTOCK TRANSACTION

{R_ID} → {Date, Administrator, ISBN, P_Name, Quantity}
{ISBN} → {P_Name}

CATEGORY

This is compound key so no functional dependency

AUTHOR:

{Author_ID} → {First Name, MI, Last Name}

WRITE:

This is compound key so no functional dependency

RATE:

{ISBN, C_Email} → {Rating}

CONSTITUTE:

{ISBN,ORDER_ID} → {O_Quantity}

ADDRESS: The ADDRESS relation is in 1NF because city and state are partially dependent on ZIP therefore it does not meet 2NF.

{Address, ZIP} → {City, State}
{ZIP} → {City, State}

C_LOCATE:

{C_Email} → {Address, ZIP}
{Address, ZIP} → {C_Email}

3. For each relation schema in your model, determine the highest normal form of the relation. If the relation is not in 3NF, rewrite your relation schema so that it is in at least 3NF.

PUBLISHER: Given that there could be multiple value entered for phone number, this relation could violate the condition of 1NF. To normalize to 3NF, we need to decompose it to two relations.

PUBLISHER
{P_Name, P_Email,P_Address, P_ZIP }

PUBLISHER_PHONE
{P_Name, P_Phone}

And actually it is now in BCNF.

BOOK: In 3NF, as every non-key attribute is fully and non-transitively dependent on the key.

BOOK
{ISBN, Title, Year, Format, Language, Price, P_Name}

Although no change is needed to achieve 3NF, due to the addition of BOOK_PUBLISHER relation below, we will take out the P_Name here to avoid redundancy.

BOOK
{ISBN, Title, Year, Format, Language, Price}

BOOK_PUBLISHER
{ISBN, P_Name}

CUSTOMER: The relation is in 1NF. For the database requirement, a customer can only have 1 phone number entered but can choose to save multiple or no credit card (C_Card) information. Therefore, we need to move the C_Card to a new relation to achieve 1NF and then the new schema would actually fulfill BCNF.

CUSTOMER_INFO
{C_Email, C_FName, C_LName, C_Phone}

CUSTOMER_PAYMENT
{C_Email, C_Card}

ADDRESS: The ADDRESS relation is in 1NF because city and state are partially dependent on ZIP therefore it does not meet 2NF. After modification below it is in BCNF.

ADDRESS
{Address, ZIP}

ZIPCODE
{ZIP, City, State}

ORDERS: This is in 3NF, since every non-key attribute is non-fully and non-transitively dependent on the key.

{Order_ID, Date, Status, PaymentInfo, C_Email}

WAREHOUSE: This is in 3NF, since every non-key attribute is non-fully and non-transitively dependent on the key.

{W_Code, W_Phone, W_Address, W_ZIP}

STORE_IN: This is in 3NF, since every non-key attribute is non-fully and non-transitively dependent on the key.

{ISBN, W_Code, Quantity}

RESTOCK TRANSACTION: This is in 2NF. Since P_Name is transitively dependent on the primary key R_ID through P_Name. To achieve 3NF, we will decompose this to two relations

RESTOCK_ID
{R_ID, Date, Admin, ISBN, Quantity}

BOOK_PUBLISHER
{ISBN, P_Name}

CATEGORY: BCNF (could be multivalued, but this is an all-key relation, which has no functional dependency)

CATEGORY
{ISBN, Category}

AUTHOR: BCNF

AUTHOR
{Author_ID, First Name, MI, Last Name}

WRITE: BCNF

WRITE
{ISBN, Author ID}

RATE: BCNF

RATE
{ISBN, C_Email, Rating}

CONSTITUTE: BCNF

CONSTITUTE
{ISBN, ORDER_ID, O_Quantity}

C_LOCATE: 3NF

{C_Email, Address, ZIP}

4. For each relation schema in your model that is in 3NF but not in BCNF, either rewrite the relation schema to BCNF or provide a short justification for why this relation should be an exception to the rule of putting relations into BCNF.

All relations after alternations are in BCNF.

5. For your database, propose at least two interesting views that can be built from your relations. These views must involve joining at least two tables together each and must include some kind of aggregation in the view. Each view must also be able to be described by a one or two sentence description in plain English. Provide the code for constructing your views along with the English language description of what the view is supposed to be providing.

1. (From Checkpoint 3 3.c)
   Find the titles and ISBNs for all books with less than 5 copies in stock

   SELECT Title, B.ISBN

```
FROM BOOK AS B, STORE_IN AS S

WHERE

        B.ISBN = S.ISBN       AND

        S.Quantity < 5 ;
```

2. (From Checkpoint 3 3.e)

   Find the total number of books purchased by a single customer (you choose how to designate the customer)

```
SELECT sum(ORDERCOUNT)

FROM ORDERCOUNT AS Oc,

(

 SELECT sum(CONSTITUTE.O_Quantity)

 FROM CUSTOMER_INFO AS Cu, ORDERS AS O, CONSTITUTE AS C

 WHERE

        Cu.C_Email = O.C_Email      AND

        O.Order_ID = C.Order_ID

) AS ORDERCOUNT

WHERE

        Oc.C_Email = 'buck.1@osu.edu'
```

3. (From Checkpoint 3 5.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 P_MAX
SELECT      Author_ID, SUM(BOOK.Price * CONSTITUTE.O_Quantity) AS
Sum_P
 FROM       BOOK AS B, CONSTITUTE AS C, WRITE AS W
WHERE       B.ISBN = C.ISBN AND C.ISBN = W.ISBN
GROUP BY    Author_ID ;


SELECT      First Name, MI, Last Name, MAX(Sum_P)
```

```
FROM        AUTHOR AS A, P_MAX AS P
WHERE       P.Author_ID = A.Author_ID;
```