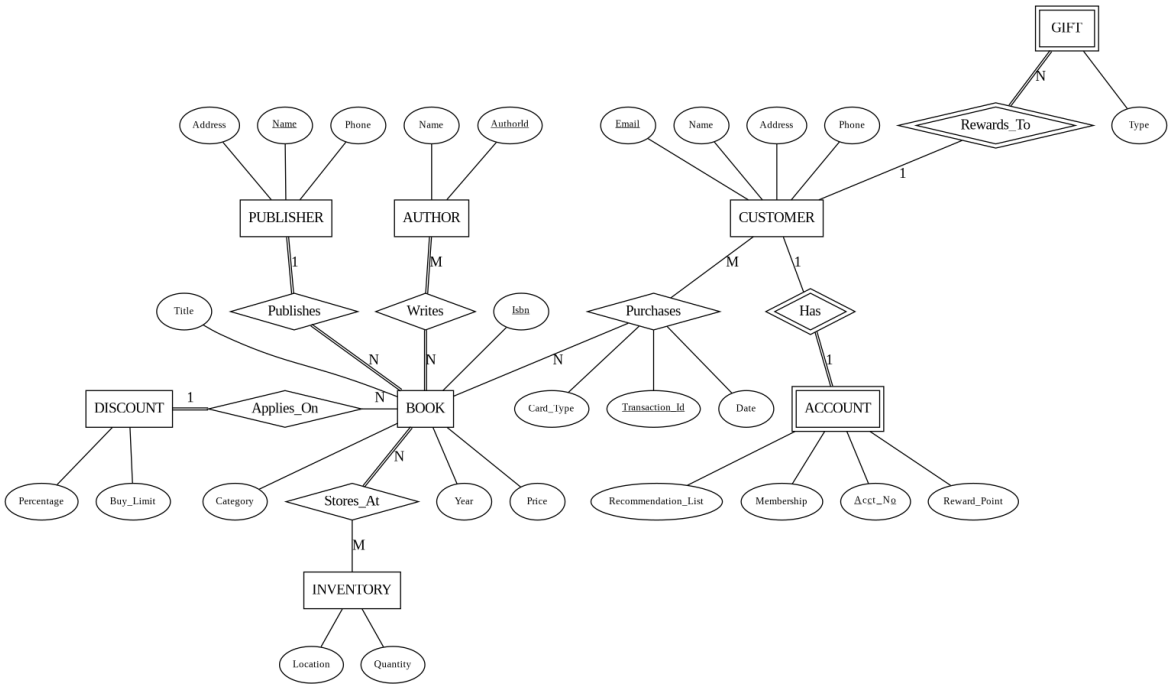
The Final Report

Wally Yang, Tina Liang, Wei Zong, Chuwen Sun, Xingyu Yan

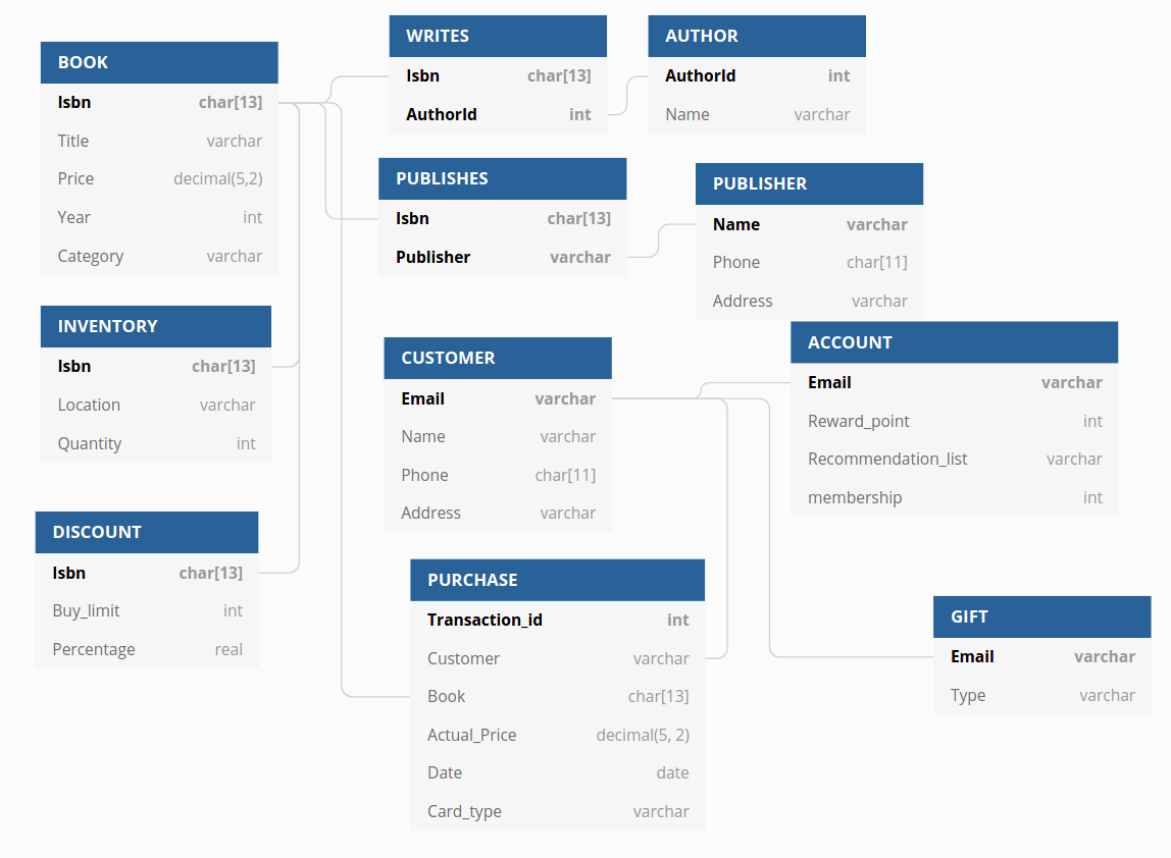
April 20, 2020

0.1 Database Description

1. ER-model of the Database Design



2. Relational Schema for the Database



The bold field refers to the primary keys of the relational schema

3. Levels of normalization for each table: All tables achieve BCNF.

4. Indices for the Database

   We chose the Tree-based index for our BOOK table since the tree-based index is good for looking up values based on range tests. It will speed up our queries when we want to retrieve the book based on the range of the year or the range of the price. Also it is not too bad for looking up values based on equality tests, so it also can slightly speed up our queries when we are trying to look up books based on the titles, categories, authors, and publishers.

5. Views for the Database

   - View A

     Description: This view is able to show all the titles and their dates of purchase made by each customer. And this could be useful to make book recommendations for a customer by looking at his or her purchase history.

Relational algebra expression:

$$R1 \leftarrow PURCHASE \bowtie_{Customer=Email} Customer$$

$$R2 \leftarrow BOOK \bowtie_{Isbn=Book} R1$$

$$Result \leftarrow \pi_{Name,\ Title,\ Date} R2$$

```
CREATE VIEW CUSTOMER_P AS
  SELECT C.Name, B.Title, P.Date
    FROM BOOK AS B, PURCHASE AS P, CUSTOMER AS C
   WHERE B.Isbn = P.Book AND P.Customer = C.Email
```

Sample output:

Luqman Finnegan	OCP: Oracle9i Certification Kit	07/01/16
Phebe Christian	SQL Server 2000 for Experienced DBA's	09/16/18
Charlie Dolan	The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling	07/20/18
Kiya Mcguire	How To Do Everything with Your Tablet PC	01/26/19
Amal Terrell	Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations	06/15/17

- View B

  Description: This view is able to show the total number of books purchased by each customer. And this could be useful to see if this customer deserves a gift by making a certain amount of purchases in this store.

  Relational algebra expression:

$$R1 \leftarrow PURCHASE \bowtie_{Customer=Email} Customer$$

$$Result \leftarrow {}_{Customer}\mathcal{F}_{COUNT\ Book}(R1)$$

```
CREATE VIEW CUSTOMER_N AS
  SELECT P.Customer, COUNT(Book)
    FROM PURCHASE AS P, CUSTOMER AS C
   WHERE P.Customer = C.Email
   GROUP BY P.Customer
```

Sample output:

Ahmed.12@osu.edu	1
Christian.2@osu.edu	1
Dolan.3@osu.edu	1
Finnegan.1@osu.edu	1
Firth.9@osu.edu	1

6. Sample Transactions for the Database

- Transaction A

  Description: The customer adds a book to a order and update the book quantity in the inventory

```sql
BEGIN TRANSACTION NEW_P;

  INSERT INTO PURCHASE
  VALUES (22,
          'Finnegan.1@osu.edu',
          '782140661',
          104.97,
          DATE(),
          'AMEX');

UPDATE Inventory
   SET Quantity = Quantity - 1
  WHERE Isbn = '782140661' AND Location = 'warehouse';

COMMIT;
```

- Transction B

  Description: A certain amount(10) of books(Isbn: 616601654) transmitted from one inventory(warehouse) to another(in-sotre)

```sql
BEGIN TRANSACTION MOVE;

UPDATE Inventory
   SET Quantity = Quantity + 10
 WHERE Isbn = '782140661' AND Location = 'warehouse';

UPDATE Inventory
   SET Quantity = Quantity - 10
 WHERE Isbn = '782140661' AND Location = 'in-store';

COMMIT;
```

- Transaction C
  Description: Customer redeem 100 reward points to a keychain

```
BEGIN TRANSACTION NEW_GIFT;

UPDATE ACCOUNT
   SET Reward_point =  Reward_point - 100
 WHERE Email = 'Finnegan.1@osu.edu';

INSERT INTO GIFT
VALUES ('Finnegan.1@osu.edu', 'keychain');

COMMIT;
```

0.2 User Manual

1. Database Description

Table	Entity	Attribute	Data type	Description	Constraints
BOOK	book	Isbn	CHAR(13)	The unique identifier of the book; The primary key	NOT NULL; fixed length of 13
		Title	VARCHAR(100)	Title of the book	NOT NULL; Up to length 100
		Price	DECIMAL(5,2)	Price of the book	NOT NULL; total 5 digits, two digits after the decimal point
		Year	INT	Year of the book got published	NOT NULL
		Category	VARCHAR(100)	Category of the book	NOT NULL; Up to length 100
INVENTORY	inventory	Isbn	CHAR(13)	The foreign key references to BOOK table; the primary key	NOT NULL; fixed length of 13
		Location	VARCHAR(100)	Location of the inventory	NOT NULL; Up to length 100
		Quantity	INT	the number of books in inventory	
DISCOUNT	discount	Isbn	CHAR(13)	The foreign key references to BOOK table	NOT NULL; fixed length of 13
		Buy_limit	INT	Purchase limit to apply the discount	
		Percentage	REAL	The discount applied to the book	NOT NULL
CUSTOMER	customer	Email	VARCHAR(100)	The unique identifier of the customer; The primary key	NOT NULL; Up to length 100
		Name	VARCHAR(100)	The name of the customer	NOT NULL; Up to length 100
		Phone	CHAR(10)	The phone of the customer	fixed length of 10
		Address	VARCHAR(100)	The address of the customer	NOT NULL; Up to length 100
AUTHOR	author	AuthorId	INT	The unique identifier of the author; The primary key	NOT NULL
		Name	VARCHAR(100)	The name of the author	NOT NULL
PUBLISHER	publisher	Name	VARCHAR(100)	The unique identifier of the publisher; The primary key	NOT NULL; Up to length 100
		Phone	CHAR(10)	The phone of the publisher	fixed length of 10
		Address	VARCHAR(100)	The address of the publisher	NOT NULL; Up to length 100
GIFT	gift	Email	VARCHAR(100)	The foreign key references to CUSTOMER table; The primary key	NOT NULL; Up to length 100
		Type	VARCHAR	The type of the gift	
ACCOUNT	account	Email	VARCHAR(100)	The foreign key references to CUSTOMER table; The primary key	NOT NULL; Up to length 100
		Reward_point	INT	The reward point of the account	
		Recommendation_list	VARCHAR(500)	The recommendation list for the customer	Up to length 500
		membership	INT	The level of the membership	
WRITES	author, book	Isbn	CHAR(13)	The foreign key references to BOOK table; One of the primary key	NOT NULL; fixed length of 13
		AuthorId	INT	The foreign key references to AUTHOR table; One of the primary key	
PUBLISHES	book, publisher	Isbn	CHAR(13)	The foreign key references to BOOK table; One of the primary key	NOT NULL
		Publisher	VARCHAR(100)	The foreign key references to PUBLISHER table; One of the primary key	NOT NULL
PURCHASE	customer, book	Transaction_id	INT	The unique identifier of the purchase; The primary key	NOT NULL
		Customer	VARCHAR(100)	The foreign key references to CUSTOMER table	NOT NULL; Up to length 100
		Book	CHAR(13)	The foreign key references to BOOK table	NOT NULL
		Date	DATE	The date of purchase	
		Card_type	VARCHAR(100)	The type of the card that customer use to purchase	Up to length 100
		Actural_Price	DECIMAL(5,2)	The price of the transaction	NOT NULL; total 5 digits, two digits after the decimal point

2. Sample SQL Queries

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

   Return a table that contains the title of that books written by Pratchett and cost less than $10

$$R1 \leftarrow BOOK * WRITES * AUTHOR$$

$$R2 \leftarrow \sigma_{Name=Terry\ Pratchett}(R1)$$

$$RESULT \leftarrow \pi_{Title,Name}(R1)$$

```sql
SELECT Title, A.Name
  FROM BOOK AS B, WRITES AS W, AUTHOR AS A
 WHERE B.Price < 10 AND A.Name LIKE 'Terry Pratchett'
   AND (B.Isbn = W.Isbn AND W.AuthorId = A.AuthorId);
```

   (b) Give all the titles and their dates of purchase made by a single customer

   Return a table that contains the title and date of the purchases made by a single customer

$$BOOKS \leftarrow BOOK \bowtie_{Isbn=Book} (\sigma_{Customer=Email}(PURCHASE))$$

$$RESULT \leftarrow \pi_{Title,Date}(BOOK)$$

```
SELECT B.Title, P.Date
  FROM BOOK AS B, PURCHASE AS P, CUSTOMER AS C
 WHERE B.Isbn = P.Book AND P.Customer = C.Email;
```

(c) Find the titles and ISBNs for all books with less than 5 copies in stock
Return a table that contains the title and ISBN for all books with less than 5 copies in inventory

$$STOCK(Isbn,\ Quantity) \leftarrow_{Isbn} \mathcal{F}_{SUM\ Quantity}(INVENTORY)$$

$$RESULT \leftarrow \pi_{Title,Isbn}(\sigma_{Quantity<5}(STOCK))$$

```
SELECT B.Title, B.Isbn
  FROM BOOK AS B, INVENTORY AS I
 WHERE B.Isbn = I.Isbn
 GROUP BY B.Title, B.Isbn
HAVING sum(I.Quantity) < 5;
```

(d) Give all the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased
Return a table that contains the customers who purchased a book by Pratchett and the titles of Pratchett books they purchased

$$R1 \leftarrow \sigma_{Name='TerryPratchett'} AUTHOR * WRITES * BOOK$$

$$R2 \leftarrow BOOK \bowtie_{Book.Isbn=PURCHASE.Book} PURCHASE$$

$$R3 \leftarrow R2 \bowtie_{PURCHASE.Customer=CUSTOMER.Email} CUSTOMER$$

$$Result \leftarrow \pi_{R2.Name,R1.Title}(R3 * R1)$$

```
SELECT C.Email, C.Name, B.Title
  FROM CUSTOMER AS C, BOOK AS B, PURCHASE AS PUR,
```

```
       WRITES AS W, AUTHOR AS A
 WHERE PUR.Book = B.Isbn AND PUR.Customer = C.Email
   AND W.AuthorId = A.AuthorId
   AND A.Name = 'Terry Pratchett';
```

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

Return a table that contains the total number of books a customer has purchased and the email of the customer

$$COUNT(Customer,\ \#\ of\ Books) \leftarrow_{Customer} \mathcal{F}_{COUNT\ BOOK}(PURCHASE)$$

$$RESULT \leftarrow \sigma_{Customer=Email}(COUNT)$$

```
SELECT P.Customer, COUNT(Book)
  FROM PURCHASE AS P, CUSTOMER AS C
 WHERE P.Customer = C.Email
GROUP BY C.Email;
```

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

Return a table that contains the customer who has purchased the most books and the amount of the books they have purchased

$$COUNT(Customer,\ No) \leftarrow_{Customer} \mathcal{F}_{COUNT\ BOOK}(PURCHASE)$$

$$RESULT \leftarrow_{Customer} \mathcal{F}_{MAX\ No}(COUNT)$$

```
SELECT L.Customer, MAX(L.Num)
  FROM PURCHASE AS P, CUSTOMER AS C,
       (SELECT P.Customer AS Customer, COUNT(Book) AS Num
          FROM PURCHASE AS P, CUSTOMER AS C
         WHERE P.Customer = C.Email
         GROUP BY P.Customer) AS L
```

(g) Find the CUSTOMER with the most Reward_point on his or her account

Return a table that contains the email and name of the customer who has the most reward point on his or her account

$$R1 \leftarrow ACCOUNT * CUSTOMER$$

$$RESULT \leftarrow_{CUSTOMER.Email, CUSTOMER.Name} \Im_{MAX\,Reward\_point}(R1)$$

```
SELECT C.Email, Name, MAX(A.Reward_point)
  FROM CUSTOMER C, ACCOUNT A
 WHERE C.Email = A.Email
```

(h) Find the most expensive BOOK with all the DISCOUNT applied

Return a table that contains ISBN and title of books that has the most expensive price with applied discount

$$DIS\_BOOKS \leftarrow BOOK \rtimes DISCOUNT$$

$$RESULT \leftarrow_{Isbn,\ Title} \mathcal{F}_{MAX(Price*percentage)}(DIS\_BOOKS)$$

```
SELECT BOOK.Isbn, Title, MAX(Price*Percentage)
  FROM (BOOK LEFT OUTER JOIN DISCOUNT)
 WHERE BOOK.Isbn = DISCOUNT.Isbn
```

(i) Find the total price of all the BOOK for each stock (quantity * price)

Return a table that contains the total price of all the books and their ISBN for each stock

$$STOCK \leftarrow BOOK *_{Isbn} \mathcal{F}_{SUM\ Quantity}(INVENTORY)$$

$$RESULT \leftarrow \pi_{Isbn, Quantity\ *\ Price}(STOCK)$$

```
SELECT Isbn, Quantity*Price
  FROM BOOK NATURAL JOIN INVENTORY
 GROUP BY Isbn
HAVING Quantity*Price=SUM(Quantity);
```

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

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

$$Customer\_spend \leftarrow CUSTOMER * ({}_{CUSTOMER}\Im_{SUM Actual\_Price}(PURCHASE))$$

$$RESULT \leftarrow \Pi_{Name,SUM(Actual\_Price)} Customer\_spend$$

```
SELECT C.Name, SUM(P.Actual_Price)
  FROM PURCHASE AS P, CUSTOMER AS C
 WHERE P.Customer = C.Email
 GROUP BY P.Customer
```

(k) Provide a list of customer names and e-mail addresses for customers who have spent more than the average customer.
Return a table that contains the names and e-mail addresses of the customers who have spent more than the average customer

$$\rho_{Total(Email,Total\_amount)}({}_{CUSTOMER}\mathcal{F}_{SUM Actual\_Price}(PURCHASE));$$

$$CUSINFO \leftarrow Total * CUSTOMER;$$

$$RESULT \leftarrow \Pi_{Name,Email}(\sigma_{Total\_amount > AVG Total\_amount}(CUSINFO))$$

```
SELECT Name, Email
  FROM CUSTOMER NATURAL JOIN (SELECT SUM Actural_Price AS Total_amount
                                FROM PURCHASE AS P
                               GROUP BY P.Customer ) AS TOTAL
HAVING TOTAL.Toal_amount > AVG (TOTAL.Total_amount)
```

(l) 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.
Return a table that contains the total number of each book that has been sold to customers, and the titles of those books, sorted by the number of books in descending number.

$$SOLD \leftarrow PURCHASE \bowtie_{Book=Isbn} BOOK$$

$$COUNT \leftarrow {}_{Isbn}\Im_{COUNTBook} SOLD$$

$$RESULT \leftarrow \Pi_{Title,COUNTBook} COUNT$$

```sql
SELECT B.Title, COUNT(P.Book)
  FROM PURCHASE AS P, BOOK AS B
 WHERE P.Book = B.Isbn
 GROUP BY P.Book
 ORDER BY COUNT(P.Book) DESC
```

(m) 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.

Return a table that contains the total dollar amount of each book that has been sold to customers, and the titles of those books, sorted by the dollar amount in descending order.

$$SOLD \leftarrow PURCHASE \bowtie_{Book=Isbn} BOOK$$

$$\rho_{Total\_book(Isbn,count)} Isbn\Im_{COUNT(Book)}(SOLD)$$

$$RESULT \leftarrow \Pi_{Title,Count*Actual\_Price}(Total\_book)$$

```sql
SELECT B.Title, COUNT(P.Book)
  FROM PURCHASE AS P, BOOK AS B
 WHERE P.Book = B.Isbn
 GROUP BY P.Book
 ORDER BY COUNT(P.Book) DESC
```

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

Return a table that contains the name of the author who sold the most books

$$SOLD \leftarrow (PURCHASE \bowtie_{Book=Isbn} BOOK) \bowtie_{Book=Isbn} WRITES$$

$$\rho_{R1(Total\_amount)}(_{AuthorId}\Im_{COUNTBook}(SOLD))$$

$$RESULT \leftarrow Name\Im_{MAX(Total\_amount)}(R1)$$

```sql
SELECT A.Name, MAX(Total)
  FROM AUTHOR AS A, (SELECT A.AuthorId, COUNT(P.Book) AS Total
                                          FROM PURCHASE AS P, BOOK AS B, WRITES
                                     WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn
```

```
                                                GROUP BY W.AuthorId) AS R1
 WHERE R1.AuthorId = A.AuthorId
 GROUP BY A.AuthorId, A.Name
```

(o) Find the most profitable author in the database for this store (i.e. the one who has brought in the most money)
Return a table that contains the name of the author who brought in the most money and the dollar amount

$$SOLD \leftarrow (PURCHASE \bowtie_{Book=Isbn} BOOK) \bowtie_{Book=Isbn} WRITES$$

$$\rho_{R1(Total\_amount)}({}_{AuthorId}\Im_{COUNTBook}(SOLD))$$

$$MAX \leftarrow \Im_{MAX(Actual\_Price*Total\_amount)}Total\_Dollar) * Total\_dollar$$

$$RESULT \leftarrow \Pi_{Name,Actual\_Price*Total\_amount}MAX$$

```
SELECT A.Name, MAX(Total)
  FROM AUTHOR AS A, (SELECT A.AuthorId, P.Actual_Price * COUNT(P.Book) AS Total
                                          FROM PURCHASE AS P, BOOK AS B, WRITES
                                    WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn
                                    GROUP BY W.AuthorId) AS R1
 WHERE R1.AuthorId = A.AuthorId
 GROUP BY A.AuthorId, A.Name
```

(p) Provide a list of customer information for customers who purchased anything written by the most profitable author in the database.
Return a table that contains the information of customers who purchased anything written by the most profitable author

$$SOLD \leftarrow (PURCHASE \bowtie_{Book=Isbn} BOOK) \bowtie_{Book=Isbn} WRITES$$

$$\rho_{R1(Total\_amount)}({}_{AuthorId}\Im_{COUNTBook}(SOLD))$$

$$Total\_dollar \leftarrow \Pi_{Name,AuthorId,Actual\_Price*Total\_amount}R1$$

$$Max\_author \leftarrow ({}_{(Name,AuthorId)}\Im_{MAX(Actual\_Price*Total\_amount)}Total\_Dollar)$$

$$CUS \leftarrow CUSTOMER \bowtie_{Email=Customer} PURCHASE$$

$$Cust\_author \leftarrow (CUS \bowtie_{Book=Isbn} WRITES) * Max\_author$$

$$RESULT \leftarrow \Pi_{Name,Email,Phone,Address}(Cust\_author)$$

```sql
SELECT C.Name, C.Email, C.Phone, C.Address
  FROM (SELECT A.AuthorId, MAX(Total)
          FROM (SELECT A.AuthorId, P.Actual_Price * COUNT(P.Book) AS Total
                  FROM PURCHASE AS P, BOOK AS B, WRITES AS W
                 WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn
                 GROUP BY W.AuthorId) AS R1,
               AUTHOR AS A
         WHERE R1.AuthorId = A.AuthorId
         GROUP BY A.AuthorId) AS R2, CUSTOMER AS C
 WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn AND W.AuthorId = R2.AuthorId
```

(q) Provide the list of authors who wrote the books purchased by the customers who have spent more than the average customer.
Return a table that contains the name of the authors whose books are purchased by the customers who have spent more than average customer

$$Author\_sold \leftarrow (WRITES * AUTHOR) \bowtie_{Isbn=Book} PURCHASE$$

$$\rho_{Spent(Email,total\_spent)}(customer\Im_{SUM Actual\_Price} PURCHASE)$$

$$CUS \leftarrow \sigma_{total\_spent>AUGtotal\_spent} Spent$$

$$RESULT \leftarrow Author\_sold \bowtie_{customer=Email} CUS$$

```sql
SELECT A.Name
  FROM AUTHOR AS A, WRITES AS W, PURCHASE AS P, CUSTOMER AS C,
       (SELECT C1.Email AS Email, sum(P1.Actual_Price) AS Spent
          FROM PURCHASE P1, CUSTOMER C1
         WHERE P1.Customer = C1.Email
         GROUP BY P1.Customer) AS personal_sum
 WHERE A.AuthorId = W.AuthorId AND W.Isbn = B.Isbn AND C.Email = personal_sum.
```

3. INSERT Syntax

```sql
-- Insert a new BOOK
INSERT INTO BOOK
VALUES ('782140661',                          -- ISBN
        'OCP: Oracle9i Certification Kit',    -- Title
        2002,                                 -- Year
        104.97,                               -- Price
        'Computer'                            -- Category
```

```
);

-- Insert the AUTHOR of the BOOK, give a unique author ID
INSERT INTO AUTHOR
VALUES (1, 'Chip Dawes');                    -- AuthorID & Author Name

INSERT INTO AUTHOR
VALUES (2, 'Biju Thomas');

INSERT INTO AUTHOR
VALUES (3, 'Doug Stuns');

INSERT INTO AUTHOR
VALUES (4, 'Matthew Weishan');

INSERT INTO AUTHOR
VALUES (5, 'Joseph C. Johnson');

-- Insert the WRITES relation between BOOK and AUTHOR

INSERT INTO WRITES
VALUES ('782140661', 1);                     -- ISBN & Author ID

INSERT INTO WRITES
VALUES ('782140661', 2);

INSERT INTO WRITES
VALUES ('782140661', 3);

INSERT INTO WRITES
VALUES ('782140661', 4);

INSERT INTO WRITES
VALUES ('782140661', 5);

-- Insert PUBLISHER information
INSERT INTO PUBLISHER
VALUES ('Sybex',                             -- Name
        '0000000000',                        -- Phone
        '123 North Ave, Columbus, Ohio, 43210'  -- Address
);

-- Insert PUBLISHES relation
INSERT INTO PUBLISHES
```

```sql
VALUES ('782140661', 'Sybex');              -- ISBN & Publisher

-- Insert BOOK discount
INSERT INTO DISCOUNT
VALUES ('782140661',                        -- ISBN
        10,                                 -- Buy Limit
        0.7                                 -- Discount Percentage
);

INSERT INTO BOOK
VALUES ('782140661',                        -- ISBN
        'OCP: Oracle9i Certification Kit',  -- Title
        2002,                               -- Year
        104.97,                             -- Price
        'Computer'                          -- Category
);

-- Insert CUSTOMER information
INSERT INTO CUSTOMER
VALUES ('brutus.1@osu.edu',                    -- Email
        'Buckeye Brutus',                      -- Name
        '0000000000',                          -- Phone
        '123 North Ave, Columbus, Ohio, 43210'  -- Address
);

-- Insert Customer ACCOUNT
INSERT INTO ACCOUNT
VALUES ('brutus.1@osu.edu',                 -- Email
        1000,                               -- Reward Point
        'Sci-fi, Novel',                    -- Recommendation List
        1                                   -- Membership
);

-- Insert GIFT information
INSERT INTO GIFT
VALUES ('brutus.1@osu.edu', 'Book')         -- Customer Email & Gift Type
```

4. DELETE Syntax

```sql
-- Delete a BOOK
DELETE FROM BOOK
 WHERE ISBN = '0782140661';

```

```
DELETE FROM AUTHOR
 WHERE (SELECT ISBN
                FROM (AUTHOR LEFT OUTER JOIN WIRTES)
          WHERE ISBN = '0782140661');

DELETE FROM WRITES
 WHERE ISBN = '0782140661';

DELETE FROM PUBLISHER
 WHERE (SELECT ISBN
                FROM (PUBLISHER LEFT OUTER JOIN PUBLISHES)
          WHERE ISBN = '0782140661');

DELETE FROM PUBLISHES
 WHERE ISBN = '0782140661';

-- Delete a PUBLISHER
DELETE FROM PUBLISHER
 WHERE Name = 'Sybex';

DELETE FROM AUTHOR
 WHERE (SELECT Authorid
                FROM WRITES
               WHERE (SELECT ISBN
                             FROM PUBLISHES
                            WHERE Publisher = 'Sybex'));

DELETE FROM WRITES
 WHERE (SELECT ISBN
                FROM PUBLISHES
               WHERE Publisher = 'Sybex');

DELETE FROM BOOK
 WHERE (SELECT ISBN
           FROM (PUBLISHERES RIGHT OUTER JOIN BOOK)
          WHERE PUBLISHER.Publisher = 'Sybex');

DELETE FROM PUBLISHES
 WHERE Publisher = 'Sybex';

-- Delete an AUTHOR
DELETE FROM AUTHOR
 WHERE AuthoreId = '12345';

```

```sql
DELETE FROM PUBLISHER
 WHERE (SELECT Publisher
                FROM PUBLISHES
               WHERE (SELECT ISBN
                             FROM WRITES
                            WHERE AuthorId='12345'));

DELETE FROM PUBLISHES
 WHERE (SELECT ISBN
                FROM WRITES
               WHERE AuthorId='12345');

DELETE FROM BOOK
 WHERE (SELECT ISBN
          FROM (WRITES RIGHT OUTER JOIN BOOK)
         WHERE AuthorId='12345');

DELETE FROM WRITES
 WHERE AuthorId='12345';

-- Delete a CUSTOMER
DELETE FROM ACCOUNT
 WHERE Email = 'email@email.com';

DELETE FROM CUSTOMER
 WHERE Email = 'email@email.com';
```

0.3 Appendix: Graded Checkpoints

0.3.1 Checkpoint 1

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, Author(s), Year, Price, Category)
   - PUBLISHER (Name, Phone, Address)
   - CUSTOMER (Id, Email, Name, Address, Phone)
   - INVENTORY (Location, Quantity)
   - SALE_RECORD (Transaction_Id, Date, Card_Type)

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

    - PUBLISHER entities publish BOOK entities
    - CUSTOMER entities purchase BOOK entities, create SALE_RECORD entities (ternary relation)
    - BOOK entities stores at INVENTORY entities

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.

    - DISCOUNT (Percentage, Buy_Limit)

      Additional Relationships: DISCOUNT entities applies on BOOK entities Applying a DISCOUNT entities to BOOK entities, the final cost of that BOOK entities could be calculated automatically.
    - ACCOUNT (weak) (Acct_No, Reward_Point, Recommendation_List, membership)

      Additional Relationships: CUSTOMER entities may have ACCOUNT entities Enabling CUSTOMER entities to have their unique account could help the stakeholders to check information of a customer quicker when the customer is trying to make a purchase.
    - GIFT weak) (Type)

      Additional Relationships: BOOKSTORE entities reward GIFT entities to CUSTOMER entities By applying GIFT entities, customers get rewarded once they spend a certain amount at the bookstore, which will maintain a good customer service.

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.

    - Report all current inventory
    - Report sales summary
    - List all current accounts with membership
    - Show all discounted books
    - List all books from certain category
    - List all books by certain author
    - List all gifts that are rewarded to customer

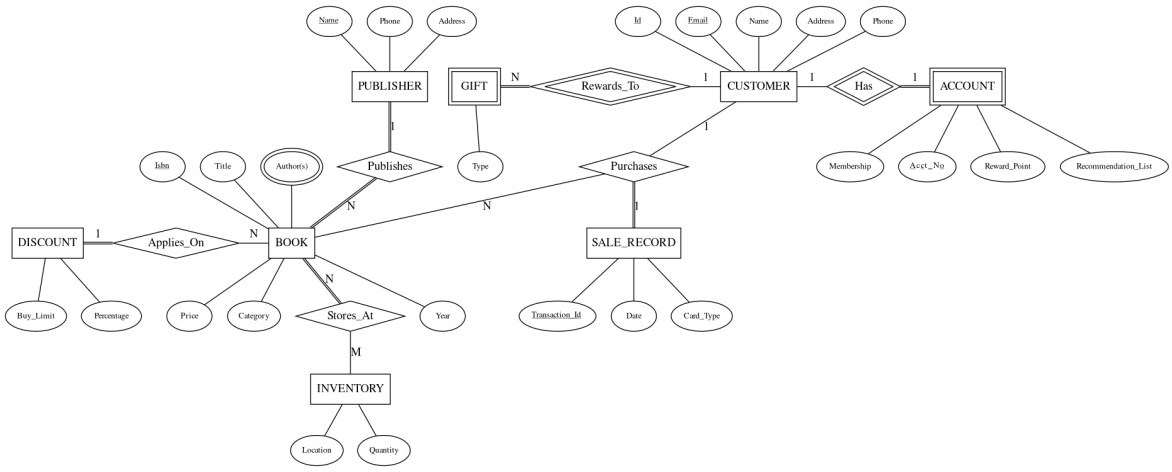
5. Suppose we want to add a new publisher to the database. How would we do that given the entities and relationships you've outlined above? Given your above description, is it possible to add a new publisher to your database without knowing the title of any books they have published? If not, revise your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

   Create PUBLISHER entities with attributes "Name", "Phone" and "Address", and have a relationship with BOOK entities which is PUBLISHER entities publish BOOK entities.

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.

   - Change the price of a BOOK
   - Update the INVENTORY of a BOOK
   - Add Reward_Point for a MEMBER
   - Change the Percentage of a DISCOUNT
   - Change the Type of GIFT
   - Customers update their personal information (Email, Name, Address, Phone)
   - Account's recommendation list can be altered according to changes in purchasing history

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.



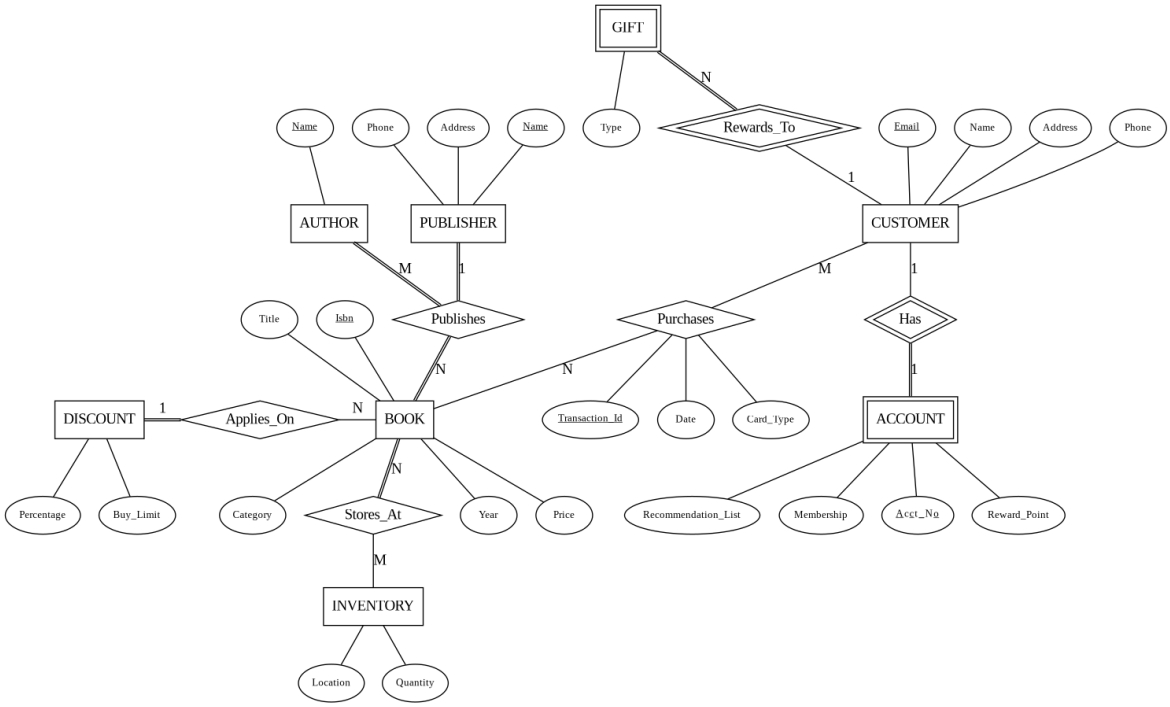
FIX:

1. Yes, we can.

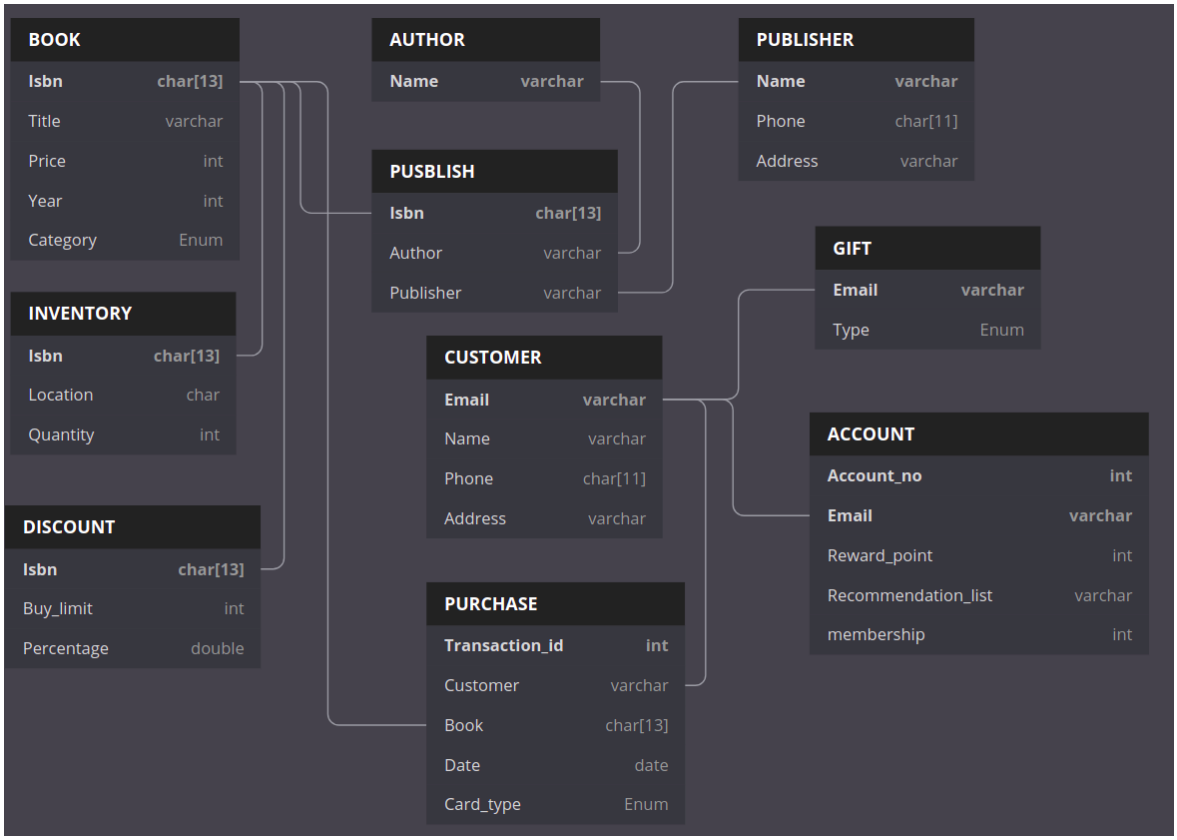
2. See section 1, page 1 for updated ER model.
3. See section 1, page 1 for updated ER model.
4. See section 1, page 1 for updated ER model.
5. ID is removed, See section 1, page 1 for updated ER model.
6. Yes, it is.

0.3.2 Checkpoint 2

1. Updated ER Model



2. Relational Schema



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

$$\pi_{Title}(\sigma_{Price<10}(BOOK))$$

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

$$BOOKS \leftarrow BOOK \bowtie_{Isbn=Book} (\sigma_{Customer=Email}(PURCHASE))$$

$$RESULT \leftarrow \pi_{Title,Date}(BOOK)$$

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

$$STOCK(Isbn,\ Quantity) \leftarrow_{Isbn} \mathcal{F}_{SUM\ Quantity}(INVENTORY)$$

$$RESULT \leftarrow \pi_{Title,Isbn}(\sigma_{Quantity<5}(STOCK))$$

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

$$PRATCHETTS \leftarrow (\sigma_{Author=Pratchett}(PUBLISH) * BOOK)$$

$$SALES \leftarrow (PRATCHETTS * PURCHASE)$$

$$RESULT \leftarrow (\pi_{Email,Name,Title}(SALES))$$

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

$$COUNT(Customer,\ \#\ of\ Books) \leftarrow_{Customer} \mathcal{F}_{COUNT\ BOOK}(PURCHASE)$$

$$RESULT \leftarrow \sigma_{Customer=Email}(COUNT)$$

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

$$COUNT(Customer,\ No) \leftarrow_{Customer} \mathcal{F}_{COUNT\ BOOK}(PURCHASE)$$

$$RESULT \leftarrow_{Customer} \mathcal{F}_{MAX\ No}(COUNT)$$

4. (a) Find the CUSTOMER with the most Reward_point on his/her account

$$CACCT \leftarrow CUSTOMER * ACCOUNT$$

$$RESULT \leftarrow \pi_{Email,\ Name}({}_{Email,\ Name}\mathcal{F}_{MAX\ Reward\_point}(CACCT))$$

(b) Find the most expensive BOOK with all the DISCOUNT applied

$$DIS\_BOOKS \leftarrow BOOK \rtimes DISCOUNT$$

$$RESULT \leftarrow_{Isbn,\ Title} \mathcal{F}_{MAX(Price*percentage)}(DIS\_BOOKS)$$

(c) Find the total price of all the BOOK for each stock (quantity * price)

$$STOCK \leftarrow BOOK *_{Isbn} \mathcal{F}_{SUM\ Quantity}(INVENTORY)$$

$$RESULT \leftarrow \pi_{Isbn,Quantity\ *\ Price}(STOCK)$$

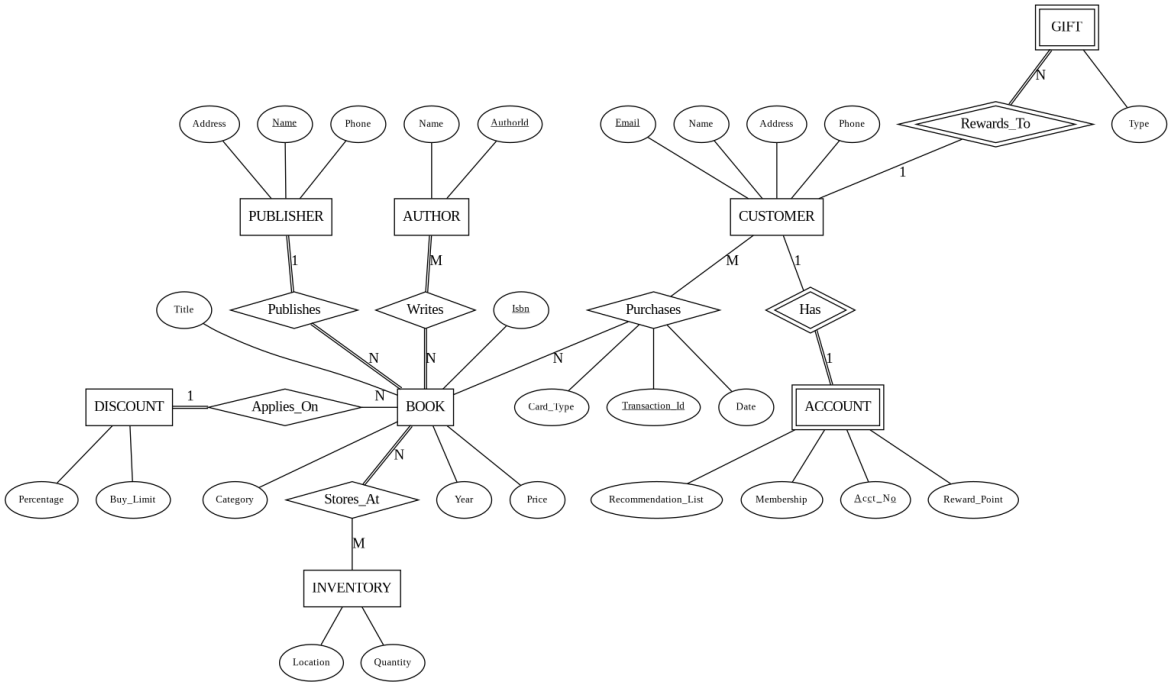
FIX:

1. See section 1, page 1 for updated ER model.
2. See section 1, page 1 for updated ER model.
3. Primary Keys in ER diagram is marked with underline
4. See section 2, page 6 for updated relational algebra
5. There might be more than one INVENTORY, so it is
6. Checked

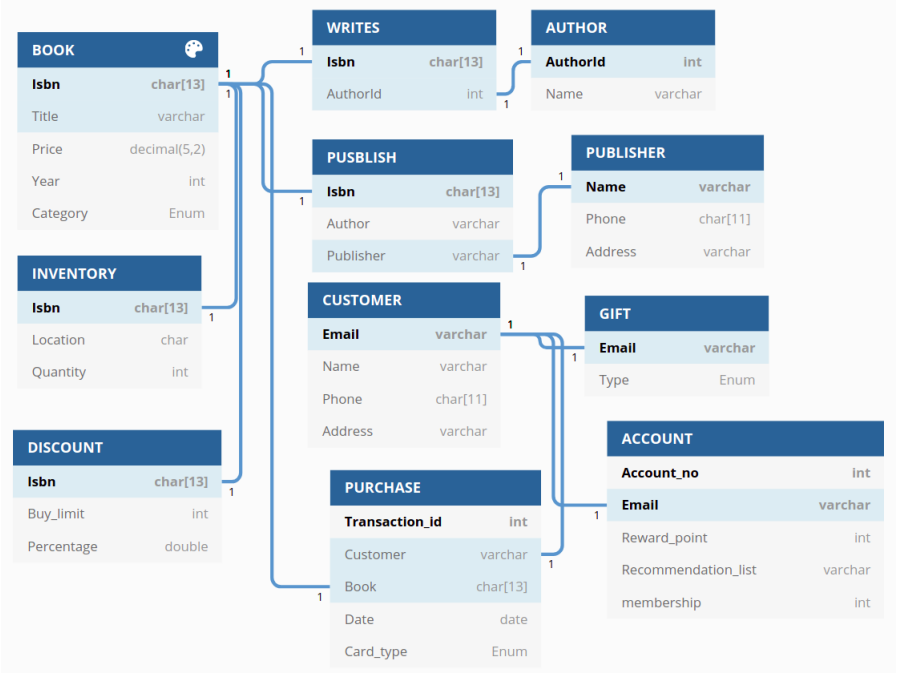
0.3.3 Checkpoint 3

Part 1

- Updated ER Model



- Updated Relational Schema



Part 2

1. Question 1

```sql
CREATE TABLE BOOK(
Isbn CHAR(13) NOT NULL,
Title VARCHAR(100) NOT NULL,
Author VARCHAR(100) NOT NULL,
Publisher VARCHAR(100) NOT NULL,
Year INT NOT NULL,
Price DECIMAL(5,2) NOT NULL,
Category VARCHAR(100) NOT NULL,
PRIMARY KEY (Isbn));

CREATE TABLE INVENTORY(
Isbn CHAR(13) NOT NULL,
Location VARCHAR(100) NOT NULL,
Quantity INT,
PRIMARY KEY (Isbn),
FOREIGN KEY (Isbn) REFERENCES BOOK(Isbn));

CREATE TABLE AUTHOR(
AuthorID INT NOT NULL,
Name VARCHAR(100) NOT NULL,
PRIMARY KEY(AuthorID));

CREATE TABLE PUBLISHER(
Name VARCHAR(100) NOT NULL,
Phone CHAR(11),
Address VARCHAR(100) NOT NULL,
PRIMARY KEY(Name));

CREATE TABLE CUSTOMER(
Email VARCHAR(100) NOT NULL,
Name VARCHAR(100) NOT NULL,
Phone CHAR(11),
Address VARCHAR(100) NOT NULL,
PRIMARY KEY(Email));

CREATE TABLE DISCOUNT(
Isbn CHAR(13) NOT NULL,
Buy_limit INT,
Percentage REAL NOT NULL,
PRIMARY KEY(Isbn),
FOREIGN KEY(Isbn) REFERENCES BOOK(Isbn));
```

```
42
43  CREATE TABLE ACCOUNT(
44  Account_no INT NOT NULL,
45  Email VARCHAR(100) NOT NULL,
46  Reward_point INT,
47  Recommendation_list VARCHAR(500),
48  Membership INT,
49  PRIMARY KEY(Account_no),
50  FOREIGN KEY(Email) REFERENCES CUSTOMER(Email));
51
52  CREATE TABLE GIFT(
53  Email VARCHAR(100) NOT NULL,
54  Type VARCHAR,
55  PRIMARY KEY(Email),
56  FOREIGN KEY(Email) REFERENCES CUSTOMER(Email));
57
58  CREATE TABLE WRITES(
59  Isbn CHAR(13) NOT NULL,
60  Author VARCHAR(100) NOT NULL,
61  PRIMARY KEY(Isbn,AuthorID),
62  FOREIGN KEY(Isbn) REFERENCES BOOK(Isbn),
63  FOREIGN KEY(AuthorID) REFERENCES AUTHOR(AuthorID));
64
65  CREATE TABLE PUBLISHES(
66  Isbn CHAR(13) NOT NULL,
67  Publisher VARCHAR(100) NOT NULL,
68  PRIMARY KEY(Isbn,Publisher),
69  FOREIGN KEY(Isbn) REFERENCES BOOK(Isbn),
70  FOREIGN KEY(Publisher) REFERENCES PUBLISHER(Name));
71
72  CREATE TABLE PURCHASE(
73  Transaction_id INT NOT NULL,
74  Customer VARCHAR(100) NOT NULL,
75  Book CHAR(13) NOT NULL,
76  Actual_Price DECIMAL(5,2) NOT NULL,
77  Date DATE,
78  Card_type VARCHAR(100),
79  PRIMARY KEY(Transaction_id),
80  FOREIGN KEY(Customer) REFERENCES CUSTOMER(Email),
81  FOREIGN KEY(Book) REFERENCES BOOK(Isbn));
```

2. Question 2

```sql
-- a
SELECT Title
FROM BOOK AS B, PUBLISHES AS P
WHERE B.Isbn = P.Isbn AND B.Price < 10 AND P.Author = 'Pratchett';

-- b
SELECT B.Title, P.Date
FROM BOOK AS B, PURCHASE AS P, C AS CUSTOMER
WHERE B.Isbn = P.Book AND P.Customer = C.Email;

-- c
SELECT B.Title, B.Isbn
FROM BOOK AS B, INVENTORY AS I
WHERE B.Isbn = I.Isbn
GROUP BY B.Title, B.Isbn
HAVING sum(I.Quantity) < 5;

-- d
SELECT C.Email, C.Name, B.Title
FROM CUSTOMER AS C, BOOK AS B, PUBLISHES AS P, PURCHASE AS PUR
WHERE P.Isbn = B.Isbn AND P.Author = 'Pratchett' AND PUR.Book = B.Isbn;

-- e
SELECT P.Customer, COUNT(Book)
FROM PURCHASE AS P, CUSTOMER AS C
WHERE P.Customer = C.Email;

-- f
SELECT L.Customer, L.Num
FROM PURCHASE AS P, CUSTOMER AS C,
(SELECT P.Customer AS Customer, COUNT(Book) AS Num
FROM PURCHASE AS P, CUSTOMER AS C
WHERE P.Customer = C.Email) AS L
WHERE L.Num = MAX(L.Num);
```

3. Question 3

```sql
SELECT Email, Name
FROM (CUSTOMER NATURAl JOIN ACCOUNT)
WHERE Reward_point=MAX(Reward_point);

SELECT Isbn, Title
FROM (BOOK LEFT OUTER JOIN DISCOUNT)
```

```sql
WHERE Percentage IS NOT NULL AND Price*Percentage=MAX(Price*Percentage);

SELECT Isbn, Stock*Price
FROM BOOK NATURAL JOIN INVENTORY
WHERE Stock=SUM(Quantity) ;
```

4. Question 4

```sql
-- a.
SELECT C.Name, SUM(P.Actual_Price)
  FROM PURCHASE AS P, CUSTOMER AS C
 WHERE P.Customer = C.Email
 GROUP BY P.Customer

-- b.
SELECT C.Name, C.Email
  FROM CUSTOMER AS C,
       (SELECT SUM(P.Actual_Price) AS Total
          FROM PURCHASE AS P
         GROUP BY P.Customer) AS R1
 WHERE R1.Customer = C.Email AND R1.Total > AVG(R1.Customer)
 GROUP BY R1.Customer

-- c.
SELECT B.Title, COUNT(P.Book)
  FROM PURCHASE AS P, BOOK AS B
 WHERE P.Book = B.Isbn
 GROUP BY P.Book
 ORDER BY COUNT(P.Book) DESC

-- d.
SELECT B.Title, P.Actual_Price * COUNT(P.Book) AS Total_dollars
  FROM PURCHASE AS P, BOOK AS B
 WHERE P.Book = B.Isbn
 GROUP BY P.Book
 ORDER BY P.Actual_Price * COUNT(P.Book) DESC

-- e.
SELECT A.Name, MAX(Total)
  FROM (SELECT A.AuthorId, COUNT(P.Book) AS Total
          FROM PURCHASE AS P, BOOK AS B, WRITES AS W
         WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn
         GROUP BY W.AuthorId) AS R1,
```

```
       AUTHOR AS A
 WHERE R1.AuthorId = A.AuthorId
 GROUP BY A.AuthorId, A.Name

-- f.
SELECT A.Name, MAX(Total)
  FROM (SELECT A.AuthorId, P.Actual_Price * COUNT(P.Book) AS Total
          FROM PURCHASE AS P, BOOK AS B, WRITES AS W
         WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn
         GROUP BY W.AuthorId) AS R1,
       AUTHOR AS A
 WHERE R1.AuthorId = A.AuthorId
 GROUP BY A.AuthorId, A.Name

-- g.
SELECT C.Name, C.Email
  FROM (SELECT A.AuthorId, MAX(Total)
          FROM (SELECT A.AuthorId, P.Actual_Price * COUNT(P.Book) AS Total
                  FROM PURCHASE AS P, BOOK AS B, WRITES AS W
                 WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn
                 GROUP BY W.AuthorId) AS R1,
                AUTHOR AS A
         WHERE R1.AuthorId = A.AuthorId
         GROUP BY A.AuthorId) AS R2, CUSTOMER AS C
 WHERE P.Book = B.Isbn AND B.Isbn = W.Isbn AND W.AuthorId = R2.AuthorId

-- h.
SELECT A.Name
FROM AUTHOR AS A, WRITES AS W, PURCHASE AS P, CUSTOMER AS C,
(SELECT C1.Email AS Email, sum(P1.Actual_Price) AS Spent
FROM PURCHASE P1, CUSTOMER C1
WHERE P1.Customer = C1.Email
GROUP BY P1.Customer) AS personal_sum
WHERE A.AuthorId = W.AuthorId
AND W.Isbn = B.Isbn
AND C.Email = personal_sum.Email
AND personal_sum.Spent > avg(personal_sum.Spent)
AND P.Customer = C.Email
AND P.Book = B.Isbn
```

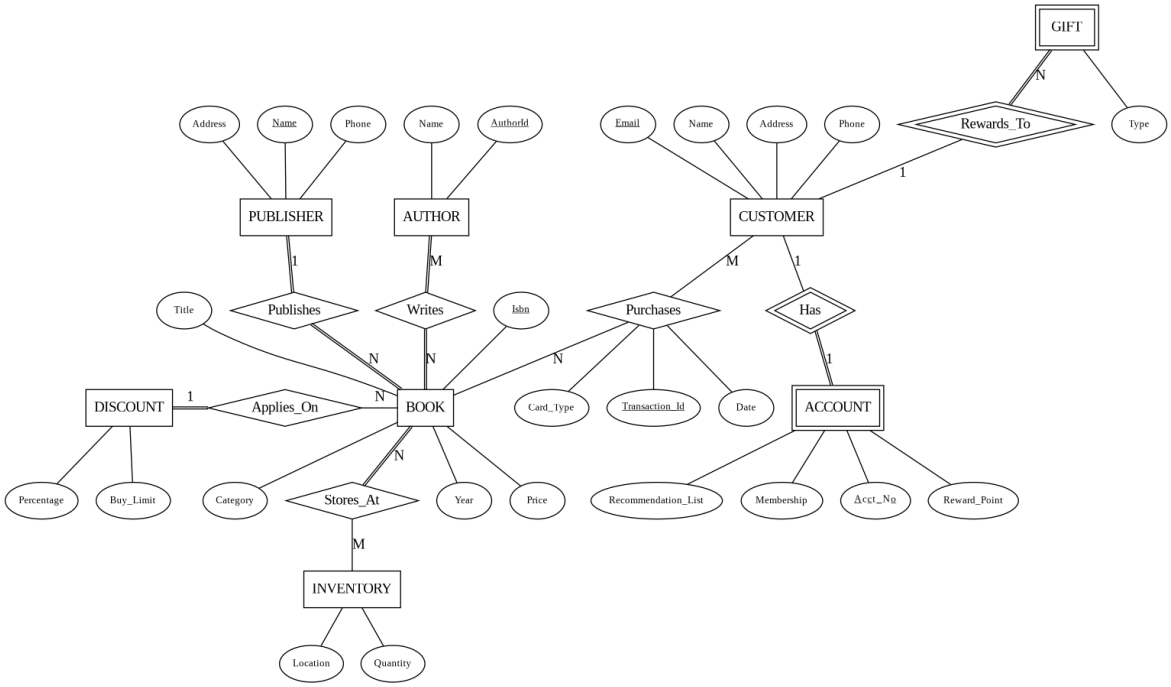
FIX:

1. See section 2, page 6 for updated SQL

2. See section 2, page 6 for updated SQL
3. See section 2, page 6 for updated SQL
4. See section 2, page 6 for updated SQL
5. Checked

0.3.4 Checkpoint 4

1. Updated ER Model



2.
   - BOOK: Isbn -> Title, Year, Price, Category
   - PUBLISHER: Name -> Address, Phone
   - AUTHOR: AuthorId -> Name
   - CUSTOMER: Email -> Name, Address, Phone
   - ACCOUNT: Customer_Email -> Membership, Reward_Point, Recommandation_List
   - PURCHASE: Transaction_Id -> Customer, Book, Card_type, Date
   - PUBLISHES: Isbn -> AuthorId, Publisher_Name
   - WRITES: Isbn -> AuthorId
3.
   - Book: BCNF
   - Publisher: BCNF
   - Author: BCNF

- Customer: BCNF
- Account: BCNF
- Purchases: BCNF
- Publishes: BCNF
- Writes: BCNF

4. N/A
5. Two Views:
   (a) View A

```
CREATE VIEW CUSTOMER_P
        AS
  SELECT B.Title, P.Date
    FROM BOOK AS B, PURCHASE AS P, C AS CUSTOMER
   WHERE B.Isbn = P.Book
```

   (b) View B

```
CREATE VIEW CUSTOMER_N
        AS
  SELECT P.Customer, COUNT(Book)
    FROM PURCHASE AS P, CUSTOMER AS C
```

FIX: N/A