

Database Management System for Bits & Books

system description and user manual

Kun Liu, Stefan Fraga, Thomas Dail, Jennifer Hong

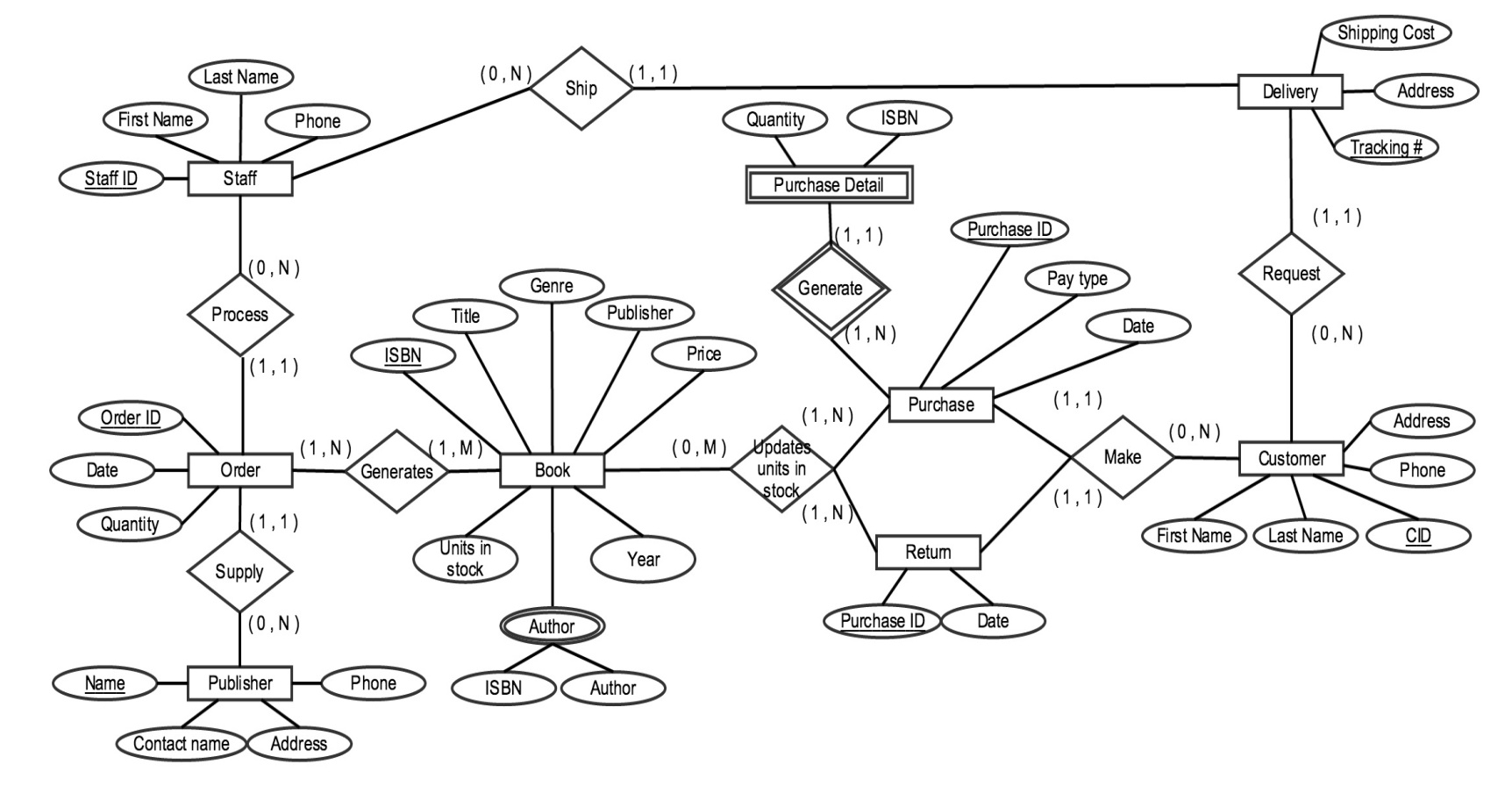
Introduction to Database Systems

CSE 3241/5241

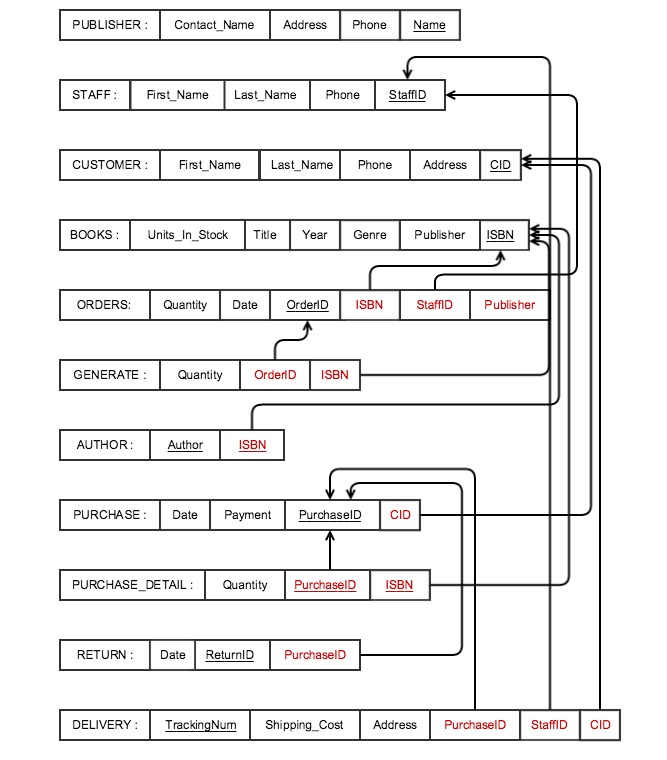
Summer 2015

**Section 1 - Database Description**

# **ER Model**



**Relational Schema**



# 

# **Normalization**

STAFF: {StaffID}→ {First\_Name, Last\_Name, DOB, Phone}

ORDER:{OrderID}→{Date, Quantity, StaffID, Publisher}

PURCHASE: {Purchase\_ID} → {Payment, Date, CID}

BOOK: {ISBN} → {Title, Publisher, year, price, genre, units\_in\_stock}

PUBLISHER: {Name} → {Contact\_Name, Address, Phone}

PURCHASE\_DETAIL:{PurchaseId, ISBN}→{Quantity}

RETURN: {Return\_ID} → {purchase\_id, Date}

DELIVERY:{Tracking\_Num} → {PurchaseID, Shipping\_cost, Address,StaffID, CID}

CUSTOMER:{CID}→{First\_Name, Last\_Name,address, phone}

GENERATE: {PurchaseID,ISBN}→{Quantity}

AUTHOR:{ISBN}→{Author}

From the dependencies above, all our relations are in BCNF.

**Index**

In our database model since we only have relatively small amount of data in the book store, we chose not to provide indexes on things other than the primary keys. Also the high update frequency nature of this database also makes frequent update if using index. There is no big difference with and without extra indexes in our data base.

**View**

**Customer View:** This view shows all books in stock from the computer genre ordered from lowest price to highest. This is useful for any customer that wants to search for books only from a specific genre and allows them to easily see what books they can afford.

SQL Statement:

SELECT BOOK.Title, BOOK.Genre, BOOK.Price

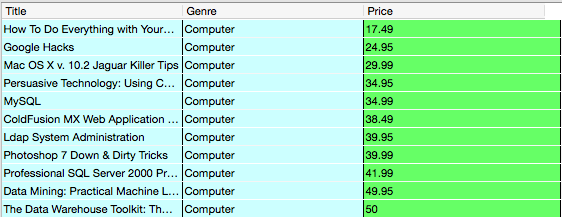
FROM BOOK

WHERE BOOK.Genre = "Computer"

GROUP BY BOOK.Price

Relational Algebra: title, genre, pricegenre=’computer’(BOOK)

View:



**Staff View:** This view allows staff members to look up the stores best sellers, it shows the title of each book and the quantity sold, ordered from most to least.

SQL Statement:

SELECT BOOK.Title, T.Quantity

FROM BOOK, (SELECT ISBN, SUM(Quantity) AS Quantity

FROM PURCHASE\_DETAIL

GROUP BY ISBN

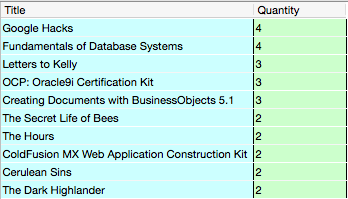
ORDER BY SUM(Quantity) DESC) AS T

WHERE BOOK.ISBN = T.ISBN;

Relational Algebra:

title, quantitygenre=’computer’(BOOK \* (ISBN, quantity)(ISBNSUM QUANTITY(Purchase\_Detail)))

View:



# **Publisher View:** This view allows a publisher to look at all books and their price from a specific publisher.

SQL Statement:

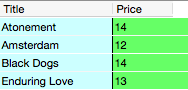
SELECT BOOK.Title, BOOK.Price

FROM BOOK

WHERE BOOK.Publisher = "Anchor Books"

Relational Algebra: title, price(publisher=”Anchor Books”(BOOK))

View:



# **Sample transactions**

1. **A customer makes a purchase of two books.**

INSERT INTO "PURCHASE" VALUES('301003325','Credit','7/3/15','244465997');

INSERT INTO "PURCHASE\_DETAIL" VALUES('782140661',1,'301003325');

INSERT INTO "PURCHASE\_DETAIL" VALUES('0061161721',1,'301003325');

UPDATE BOOK

SET Units\_in\_stock = Units\_in\_stock - 1

WHERE ISBN='782140661' and ISBN='0061161721';

**2) The same customer returns the books.**

INSERT INTO "RETURN" VALUES('65821492','301003325','7/7/15');

UPDATE BOOK

SET Units\_in\_stock = Units\_in\_stock + 1

WHERE ISBN='782140661' and ISBN='0061161721';

**3) An employee makes an order from a publisher.**

INSERT INTO "ORDERS" VALUES('704993087','5/5/15',36,'471561858','John Wiley & Sons','471363049');

UPDATE BOOK

SET Units\_in\_stock = Units\_in\_stock + 36

WHERE ISBN='782140661' and ISBN='471363049';

**Section 2 - User Manual**

# **Attribute description**

**Staff:** This table represents the actual employees of Bits and Books.

* StaffID: A unique identification number for each employee.
* First Name: Employee’s first name.
* Last Name: Employee’s last name.
* Phone: A cell phone or home phone where the employee can be reached.

**Orders:** Table containing every order made by an employee from a publisher.This only contains transactions between employees and publishers

* OrderID: A unique identification number for each order.
* Date: Date that employee ordered books from publisher
* Quantity: Number of the books that employee ordered.

**Publisher:** Table contains publishers who provide books to Bits and Books.

* Name: Publisher’s Name.
* Contact\_name: Name of the contact person that Bits and books contact for order.
* Address: Publisher’s address.
* Phone: Publisher’s phone number which will directly connects with contact person.

**Book:** Table containing the books that Bits and Books currently have.

* ISBN: Book’s ISBN.
* Title: Book’s title.
* Genre: Book’s genre.
* Publisher: Publisher of the books.
* Price: Price of the book.
* Year: Year of book’s first edition published.
* Units\_in\_Stock: Unit number of the book that Bits and Books have in stock.

**Purchase:** Table of purchase made by customer to buy books.

* PurchaseID: A unique identification number for each purchase.
* Payment: Type of payment that customer make during purchase.
* Date: Date of purchase.

**Purchase\_detail:** Details of purchase that customer made.

* Quantity: Number of the books that purchase have.
* ISBN: Book’s ISBN

**Return:** Table of returns that customer made.

* ReturnID: A unique identification number for each return.
* Date: Date of the return.

**Delivery:** Table of the delivery that requested by the customers and shipped by the staff.

* Shipping\_Cost: Shipping cost for the delivery.
* Address : Address of customer.
* TrackingNum : Tracking number for the delivery

**Customer:** Table of the customer that

* CID: Unique identification number for each customer.
* First\_Name:Customer’s first name.
* Last\_Name:Customer’s last name.
* Phone: Customer’s phone number.
* Address: Customer’s address.

**Author:** Table of authors of the book.

* ISBN: ISBN of the book that author wrote.
* Author: Name of the author.

# **Sample SQL queries**

The following sample SQL queries are provided to better explain the functionality of the data base. Each query will include:

i. An English language description of what the query should be returning

ii. The correct relational algebra syntax of the query

iii. The equivalent SQL query

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

πTitle(𝜎 author = Patchett (AUTHOR) ⋈ ISBN=ISBN (Price < 10 (BOOK))

SELECT Title

FROM BOOK AS B, AUTHOR AS A

WHERE B.ISBN=A.ISBN AND A.Author = 'Terry Pratchett' AND B.Price < 10;

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

πTitle, date((BOOK \* PURCHASE\_DETAIL) \* (𝜎 CID=244465997(PURCHASE)))

SELECT B.Title, P.Date

FROM BOOK AS B, PURCHASE AS P, PURCHASE\_DETAIL AS PD

WHERE B.ISBN=PD.ISBN AND P.PurchaseID=PD.PurchaseID AND P.CID='244465997';

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

πTitle, IBSN (𝜎Units\_in\_stock < 5 (BOOK))

SELECT Title, ISBN

FROM BOOK

WHERE Units\_in\_Stock < 5;

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

πname, title ((𝜎 Author = Pratchett (AUTHOR))\*BOOK\*(PURCHASE\_DETAIL)\*(PURCHASE))

SELECT C.First\_Name, C.Last\_Name, B.Title

FROM CUSTOMER AS C, BOOK AS B, AUTHOR AS A, PURCHASE AS P, PURCHASE\_DETAIL AS PD

WHERE C.CID=P.CID AND B.ISBN=A.ISBN AND A.ISBN=PD.ISBN AND P.PurchaseID=PD.PurchaseID AND A.Author='Terry Pratchett';

**e. Find the total number of books purchased by a single customer with CID=244465997**

ℑsum Quantity (𝜎CID=244465997 (PURCHASE\_DETAIL))

SELECT sum(Quantity)

FROM PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE PD.PurchaseID=P.PurchaseId AND P.CID='244465997';

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

π name, sum\_quantity (ℑmax sum\_quantity (CID Fsum Quantity (PURCHASE\_DETAIL)⋈CID=CID(CUSTOMER)))

SELECT Fname, Lname, max(TotalQuantity)

FROM (SELECT C.First\_Name AS Fname, C.Last\_Name AS Lname, sum(PD.Quantity) AS TotalQuantity

FROM CUSTOMER AS C, PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE C.CID=P.CID AND P.PurchaseID=PD.PurchaseID

GROUP BY C.CID);

**g. Find returns made by one customer (ID = 244465997), return date and IBSN.**

πDate, IBSN(𝜎CID=244465997 (PURCHASE)\*(PURCHASE\_DETAIL)\*(RETURN))

SELECT R.Date, PD.ISBN

FROM RETURN AS R, PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE R.PurchaseID=P.PurchaseID AND P. PurchaseID=PD.PurchaseID AND P.CID='244465997';

**h. Check states of delivery of one purchase (PurchaseID=301003325)**

πTrackingNum(𝜎PID=301003325 (Delivery))

SELECT TrackingNum

FROM DELIVERY

WHERE PurchaseID ='301003325';

**i. Find publisher name that provides most orders and calculate total price.**

πname, sum\_total (ℑmax sum\_total(PubID Fsum total\_price(Order)))⋈PubID(Publisher))

SELECT Pname, max(Sum)

FROM (SELECT sum(O.Quantity) AS Sum, P.Name AS Pname

FROM PUBLISHER AS P, ORDERS AS O

WHERE P.Name=O.Publisher

GROUP BY P.Name);

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

πFirst\_Name, Last\_Name, Price (C .CIDℑ sum Price(BOOK\*PURCHASE\_DETAIL\*PURCHASE\*CUSTOMER))

SELECT C.First\_Name, C.Last\_Name, sum(B.Price)

FROM Customer AS C, Purchase AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE P.CID=C.CID AND PD. PurchaseID=P.PurchaseID AND B.ISBN=PD.ISBN

GROUP BY C.CID;

**k. Provide a list of customer names and addresses for customers who have spent more than the average customer.**

πFirst\_Name, Last\_Name, Address (ℑavg (C .CIDF sum Price(BOOK\*PURCHASE\_DETAIL\*PURCHASE\*CUSTOMER)))

SELECT C.First\_Name, C.Last\_Name, C.Address

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE P.CID=C.CID AND P.PurchaseID=PD.PurchaseID AND PD.ISBN=B.ISBN

GROUP BY C.CID

HAVING sum(B.Price) >

(SELECT avg (Total)

FROM (SELECT sum(B.Price) AS Total

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE B.ISBN=PD.ISBN AND P.CID=C.CID AND P.PurchaseID=PD.PurchaseID

GROUP BY C.CID));

**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.**

πTitle, Quantity(ISBN ℑsum Quantity (PURCHASE\_DETAIL)\*(BOOK))

SELECT Title, sum(Quantity)

FROM BOOK AS B, PURCHASE\_DETAIL AS P

WHERE B.ISBN=P.ISBN

GROUP BY P.ISBN

ORDER BY 2 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.**

πTitle, Price\*Quantity (ISBN ℑsum Price\*Quantity (PURCHASE\_DETAIL)\*(BOOK))

SELECT Title, sum(B. Price\*P. Quantity) AS DollarTotal

FROM Book AS B, Purchase\_Detail AS P

WHERE B.ISBN=P.ISBN

GROUP BY P.ISBN

ORDER BY 2 DESC;

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

R1 🡨 Author ℑsum Quantity (PURCHASE\_DETAIL) \* (AUTHOR)

Result 🡨πAuthor (ℑmax Quantity (R1))

SELECT Name, max(sum)

FROM(

SELECT A.AUTHOR AS Name, sum(TotalQuantity) AS sum

FROM Author AS A, (

SELECT ISBN, sum(Quantity) AS TotalQuantity

FROM Purchase\_Detail

GROUP BY ISBN) AS S

WHERE A.ISBN=S.ISBN

GROUP BY A. AUTHOR);

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

R1 🡨Authorℑsum QuantityxPrice (PURCHASE\_DETAIL) \* (AUTHOR)\*(BOOK)

Result 🡨πAuthor (ℑmax QuantityxPrice (R1))

SELECT Name, max(DollarTotal)

FROM(

SELECT sum(B.Price\*PD.Quantity) AS DollarTotal, A.AUTHOR AS Name

FROM AUTHOR AS A, BOOK AS B, PURCHASE\_DETAIL AS PD

WHERE B.ISBN=A.ISBN AND B.ISBN=PD.ISBN

GROUP BY A.AUTHOR);

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

R1 🡨Authorℑsum QuantityxPrice (PURCHASE\_DETAIL) \* (AUTHOR)\*(BOOK)

R2 🡨πAuthor (ℑmax QuantityxPrice (R1))

πFirst\_Name, Last\_Name, Address ((R2)\*(PURCHASE\_DETAIL)\*(PURCHASE)\*(CUSTOMER))

SELECT C.CID ,C.First\_Name, C.Last\_Name, C.Address

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, AUTHOR AS A, BOOK AS B,

(SELECT Name, max(DollarTotal)

FROM(

SELECT sum(B.Price\*PD.Quantity) AS DollarTotal, A.Author AS Name

FROM AUTHOR AS A, BOOK AS B, PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE B.ISBN=A.ISBN AND P.PurchaseID=PD.PurchaseID AND B.ISBN=PD.ISBN

GROUP BY A.AUTHOR)) AS M

WHERE M.Name=A.AUTHOR AND A.ISBN=PD.ISBN AND PD.PurchaseID=P.PurchaseID AND C.CID=P.CID

GROUP BY C.CID;

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

R1 🡨 πISBN (ℑavg (C .CIDF sum Price(BOOK\*PURCHASE\_DETAIL\*PURCHASE\*CUSTOMER)))

R2 🡨πAuthor (R1)\*(AUTHOR)

SELECT A.Author

FROM (

SELECT B.ISBN AS ISBN

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE P.CID=C.CID AND P.PurchaseID=PD.PurchaseID AND PD.ISBN=B.ISBN

GROUP BY C.CID

HAVING sum(B.Price) >

(SELECT avg (Total)

FROM (SELECT sum(B.Price) AS Total

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE B.ISBN=PD.ISBN AND P.CID=C.CID AND P.PurchaseID=PD.PurchaseID

GROUP BY C.CID))) AS S, AUTHOR AS A

WHERE A.ISBN=S.ISBN

GROUP BY A.Author;

**INSERT syntax**

INSERT INTO "PUBLISHER" VALUES('Sybex','Diana','(844) 424-6122','Mill Lane, Hinton Saint George, Somerset TA17 8SZ, UK');

INSERT INTO "BOOK" VALUES('782140661','OCP: Oracle9i Certification Kit','Sybex',2002,'$104.97 ','Computer',62);

INSERT INTO "AUTHOR" VALUES('782140661','Biju Thomas');

INSERT INTO "CUSTOMER" VALUES('244465997','Linda','Adams','(822) 396-2864','Gold Corner Drove, Highbridge, Somerset TA9, UK');

The publisher is inserted first. The book is inserted second because it holds a foreign key that references the publisher. Author is inserted last because it holds the ISBN as a foreign key that references book. Customer may be inserted at any time.

# **DELETE syntax**

Delete Author: DELETE FROM AUTHOR WHERE Author='Biju Thomas';

Delete Book: DELETE FROM AUTHOR WHERE ISBN='782140661';

DELETE FROM BOOK WHERE ISBN='782140661';

Delete Publisher: DELETE FROM PUBLISHER WHERE Name='Sybex';

Delete Customer: DELETE FROM CUSTOMER WHERE CID='244465997';

Delete Delivery: DELETE FROM DELIVERY WHERE TrackingNum='4722262791';

Delete Generate: DELETE FROM GENERATE WHERE OrderID='704993087';

Delete Order: DELETE FROM ORDERS WHERE OrderID='704993087';

Delete Purchase: DELETE FROM PURCHASE WHERE PurchaseID='301003325';

Delete Purchase\_Detail: DELETE FROM PURCHASE\_DETAIL WHERE PurchaseID='30100325' and ISBN='782140661';

Delete Return: DELETE FROM RETURN WHERE ReturnID='65821492';

Delete Staff: DELETE FROM STAFF WHERE StaffID='471561858';

**Section 3 - Original Checkpoint Documents**

# **Checkpoint 1**

-**Entities and Relationships**

In a **NEATLY TYPED** document, provide the following:

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.

Staff: Staff ID, Name (first, last), DOB, Address, Phone, username, password

Publisher: Publisher ID, Name, Contact Name(first, last), Address, Phone, Fax, Email

Customer: Customer ID, Name (first, last), Address, Phone, Email

Order: Order ID, Staff ID, date, unit price, quantity, total price

Purchase: Purchase ID, date, list of ISBN, total price, payment type

Book: ISBN(10, 13), Book title, Publisher ID, unit price, units in stock, units pending, shelf ID

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

STAFF entities place ORDER entities

PUBLISHER entities supply ORDER entities

STAFF entities order from PUBLISHER entities.

ORDER entities generate BOOK entities.

CUSTOMER entities make PURCHASE entities.

PURCHASE entities update units in stock of BOOK entities.

1. 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.

Return: Purchase ID, total price, customer ID,

Delivery: Order ID, tracking number, Shipping address, cost

CUSTOMER entities make RETURN entities.

CUSTOMER entities request DELIVERY entities.

STAFF entities complete DELIVERY entities.

RETURN entities allow for better customer satisfaction.

DELIVERY entities allow online ordering and delivery and expand your market.

1. 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.

1) Get the list of best selling books

2) Calculate the monthly revenue of sales by adding up total payment received and subtracting returns

3) Check inventory of a book and make orders from publisher

4) Check the reason of return of an order

5) checking the status of online ordering and delivery

1. 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.

Add a new publisher under the PUBLISHER entity by creating a new publisher ID and entering all the attributes information. Yes it is possible to add a new publisher to our database without knowing the title of any books they have published.

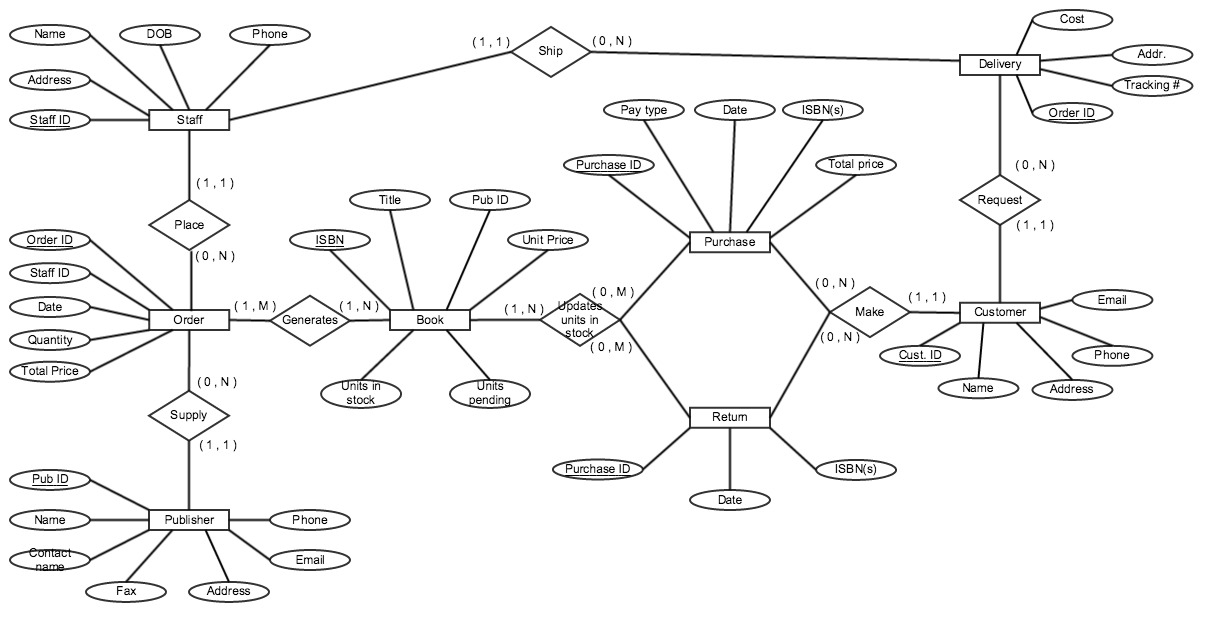
1. Determine at least three other informal update operations and describe what entities would need to have attributes altered and how they would need to be changed given your above descriptions. Include one example for each of the additional entities you proposed in question 3 above.

1) Hiring a new staff: create a new object in Staff entity including new Staff ID and other attributes.

2) Make a return: Return entity creates a new return with return ID and other attributes, after the return is completed, the BOOK entity will update unit in stock.

3) Place an order online: Order entity creates a new order with order ID and other attributes, also a bill number is created to open up a new payment. Once the payment is processed, the BOOK entity will update unit in stock. Then the DELIVERY entity will create a new delivery, staff will update the delivery by adding shipping date, tracking number. Once delivered, the delivered date will be added.

1. 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.



# **Checkpoint 2**

**– Relational Model and Relational Algebra**

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.

See attached.

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

See attached.

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

πtitle(𝜎 author = pratchett AND UnitPrice < 10 (BOOK))

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

πtitle, date(BOOK ⋈ IBSN=IBSN (𝜎cid=674533 (PURCHASE\_DETAIL)) ⋈Purchase\_ID=PID(PURCHASE)

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

πTitle, IBSN (𝜎U\_in\_stock < 5 (BOOK))

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

πname, title ((𝜎 author = Pratchett (BOOK))⋈IBSN=IBSN(PURCHASE)⋈CID=CID(CUSTOMER))

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

Assume customer ID = 674533

Fsum quantity (𝜎CID=674533(Purchase\_detail))

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

πname, sum\_quantity (Fmax sum\_quantity (CID Fsum Quantity (Purchase\_detail ⋈CID=CUST\_ID(customer)))

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.

(1)Find returns made by one customer (ID = 674533), return date and IBSN.

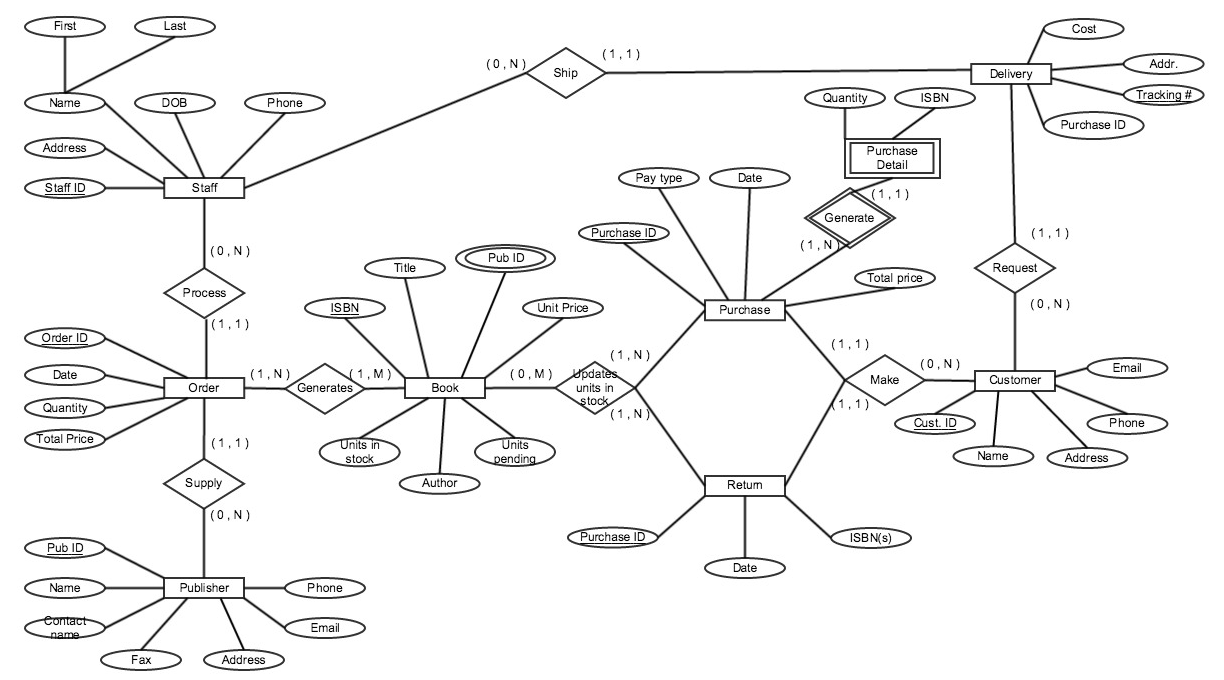
πDate, IBSN(𝜎CID=674533(Return))

(2) Check states of delivery of one purchase(purchase\_id=785678

πtracking#(𝜎PID=785678(Delivery))

(3)Find publisher name that provides most orders and calculate total price.

πname, sum\_total (Fmax sum\_total(PubID Fsum total\_price(Order)))⋈PubID(Publisher))



Staff ( Staff ID, Address, Name, DOB, Phone )

Order ( Order ID, Date, Quantity, Total Price, Staff ID, Pub ID )

Publisher ( Pub ID, Name, Contact Name, Fax, Address, Email, Phone )

Book ( ISBN, Title, Pub ID, Unit Price, Units in Stock, Units Pending)

Purchase ( Purchase ID, Pay type, Date, ISBN, Quantity, Total Price, CID\*\* )

Purchase\_Detail ( ISBN, Quantity, Purchase ID, CID )

Return ( Purchase ID, Date, ISBN, CID )

Delivery ( tracking #, Purchase ID, Cost, Addr., Staff ID, CID )

Customer ( CID, Name, Addr., Phone, Email )

Generate ( Order ID, ISBN )

Units\_in\_stock ( ISBN, Purchase ID )

Author(ISBN, Name)

# **Checkpoint 3**

1. 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”:
   1. Find the titles of all books by Pratchett that cost less than $10

SELECT Title

FROM BOOK AS B, AUTHOR AS A

WHERE B.ISBN=A.ISBN AND Author = 'Terry Pratchett' AND B.Price < 10;

* 1. 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 PURCHASE, BOOK, PURCHASE\_DETAIL

WHERE PURCHASE.CID = '171592076' AND PURCHASE\_DETAIL.ISBN = BOOK.ISBN AND PURCHASE\_DETAIL.PurchaseID = PURCHASE.PurchaseID;

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

SELECT Title, ISBN

FROM BOOK

WHERE Units\_in\_Stock < 5;

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

SELECT C.First\_Name, C.Last\_Name, B.Title

FROM CUSTOMER AS C, BOOK AS B, AUTHOR AS A, PURCHASE AS P, PURCHASE\_DETAIL AS PD

WHERE C.CID=P.CID AND B.ISBN=A.ISBN AND A.ISBN=PD.ISBN AND P.PurchaseID=PD.PurchaseID AND A.Author='Terry Pratchett'

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

ℑsum Quantity (𝜎CID=244465997 (PURCHASE\_DETAIL))

SELECT sum(Quantity)

PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE PD.PurchaseID=P.PurchaseId AND P.CID='244465997'

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

SELECT Fname, Lname, max(TotalQuantity)

FROM (SELECT C.First\_Name AS Fname, C.Last\_Name AS Lname, sum(PD.Quantity) AS TotalQuantity

FROM CUSTOMER AS C, PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE C.CID=P.CID AND P.PurchaseID=PD.PurchaseID

GROUP BY C.CID);

1. 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”.

(1)Find returns made by one customer (ID = 244465997), return date and IBSN.

SELECT R.Date, PD.ISBN

FROM RETURN AS R, PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE R.PurchaseID=P.PurchaseID AND P. PurchaseID=PD.PurchaseID AND P.CID='244465997';

(2) Check states of delivery of one purchase (PurchaseID=301003325)

SELECT TrackingNum

FROM DELIVERY

WHERE PurchaseID ='301003325'

(3)Find publisher name that provides most orders and calculate total price.

SELECT Pname, max(Sum)

FROM (SELECT sum(O.Quantity) AS Sum, P.Name AS Pname

FROM PUBLISHER AS P, ORDERS AS O

WHERE P.Name=O.Publisher

GROUP BY P.Name);

1. 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”.
   1. Provide a list of customer names, along with the total dollar amount each customer has spent.

SELECT C.First\_Name, C.Last\_Name, sum(B.Price)

FROM Customer AS C, Purchase AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE P.CID=C.CID AND PD. PurchaseID=P.PurchaseID AND B.ISBN=PD.ISBN

GROUP BY C.CID

* 1. Provide a list of customer names and addresses for customers who have spent more than the average customer.

SELECT C.First\_Name, C.Last\_Name, C.Address

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE P.CID=C.CID AND P.PurchaseID=PD.PurchaseID AND PD.ISBN=B.ISBN

GROUP BY C.CID

HAVING sum(B.Price) >

(SELECT avg (Total)

FROM (SELECT sum(B.Price) AS Total

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE B.ISBN=PD.ISBN AND P.CID=C.CID AND P.PurchaseID=PD.PurchaseID

GROUP BY C.CID));

* 1. 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, sum(Quantity)

FROM BOOK AS B, PURCHASE\_DETAIL AS P

WHERE B.ISBN=P.ISBN

GROUP BY P.ISBN

ORDER BY 2 DESC;

* 1. 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(B. Price\*P. Quantity) AS DollarTotal

FROM Book AS B, Purchase\_Detail AS P

WHERE B.ISBN=P.ISBN

GROUP BY P.ISBN

ORDER BY 2 DESC;

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

SELECT Name, max(sum)

FROM(

SELECT A.AUTHOR AS Name, sum(TotalQuantity) AS sum

FROM Author AS A, (

SELECT ISBN, sum(Quantity) AS TotalQuantity

FROM Purchase\_Detail

GROUP BY ISBN) AS S

WHERE A.ISBN=S.ISBN

GROUP BY A. AUTHOR)

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

SELECT Name, max(DollarTotal)

FROM(

SELECT sum(B.Price\*PD.Quantity) AS DollarTotal, A.AUTHOR AS Name

FROM AUTHOR AS A, BOOK AS B, PURCHASE\_DETAIL AS PD

WHERE B.ISBN=A.ISBN AND B.ISBN=PD.ISBN

GROUP BY A.AUTHOR);

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

SELECT C.CID ,C.First\_Name, C.Last\_Name, C.Address

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, AUTHOR AS A, BOOK AS B,

(SELECT Name, max(DollarTotal)

FROM(

SELECT sum(B.Price\*PD.Quantity) AS DollarTotal, A.Author AS Name

FROM AUTHOR AS A, BOOK AS B, PURCHASE\_DETAIL AS PD, PURCHASE AS P

WHERE B.ISBN=A.ISBN AND P.PurchaseID=PD.PurchaseID AND B.ISBN=PD.ISBN

GROUP BY A.AUTHOR)) AS M

WHERE M.Name=A.AUTHOR AND A.ISBN=PD.ISBN AND PD.PurchaseID=P.PurchaseID AND C.CID=P.CID

GROUP BY C.CID

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

SELECT A.Author

FROM (

SELECT B.ISBN AS ISBN

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE P.CID=C.CID AND P.PurchaseID=PD.PurchaseID AND PD.ISBN=B.ISBN

GROUP BY C.CID

HAVING sum(B.Price) >

(SELECT avg (Total)

FROM (SELECT sum(B.Price) AS Total

FROM CUSTOMER AS C, PURCHASE AS P, PURCHASE\_DETAIL AS PD, BOOK AS B

WHERE B.ISBN=PD.ISBN AND P.CID=C.CID AND P.PurchaseID=PD.PurchaseID

GROUP BY C.CID))) AS S, AUTHOR AS A

WHERE A.ISBN=S.ISBN

GROUP BY A.Author

# **Checkpoint 4**

**– Functional Dependencies and Normal Forms**

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.**

**Staff ( Staff ID, First\_Name, Last\_Name, Phone)**

**Orders ( Order ID, Date, Quantity, Staff ID, Publisher, ISBN)**

**Publisher ( Name, Contact\_Name, Address, Phone)**

**Book ( ISBN, Title, Publisher, Year, Price, Genre, Units\_in\_Stock)**

**Purchase ( Purchase ID, Payment, Date, CID )**

**Purchase\_Detail ( ISBN, Quantity, Purchase ID)**

**Return ( Return ID, Purchase\_ID ,Date)**

**Delivery ( TrackingNum, Shipping\_Cost, Address, Purchase ID, Staff ID, CID )**

**Customer ( CID, First\_Name, Last\_Name, Address., Phone)**

**Generate ( Order\_ ID, ISBN , Quantity)**

**Author(ISBN, Name)**

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).

STAFF: {StaffID}→ {First\_Name, Last\_Name, DOB, Phone}

ORDER:{OrderID}→{Date, Quantity, StaffID, Publisher}

PURCHASE: {Purchase\_ID} → {Payment, Date, CID}

BOOK: {ISBN} → {Title, Publisher, year, price, genre, units\_in\_stock}

PURCHASE: {Purchase\_ID} → {Payment, Date, CID}

PURCHASE\_DETAIL:{PurchaseId, ISBN}→{Quantity}

RETURN: {Return\_ID} → {purchase\_id, Date}

DELIVERY:{Tracking\_Num} → {PurchaseID, Shipping\_cost, Address,StaffID, CID}

CUSTOMER:{CID}→{First\_Name, Last\_Name,address, phone}

GENERATE: {PurchaseID,ISBN}→{Quantity}

AUTHOR:{ISBN}→{Author}

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.

All of our relations are in 3NF.

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 of our relations 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.

-Create a view of publisher contact names and phone numbers that have total of more than 100 books in stock in store

SELECT P.Contact\_Name, P.Phone

FROM PUBLISHER AS P, BOOK AS B

WHERE B.Publisher=P.Name

GROUP BY P.Name

HAVING sum(B.Units\_in\_Stock>100);

-Find customer names who had returned more than 10 purchases:

SELECT C.First\_Name, C.Last\_Name

FROM CUSTOMER AS C, RETURN AS R, PURCHASE AS P

WHERE P.PurchaseID=R.PurchaseID AND C.CID=P.CID

GROUP BY C.CID

HAVING count (R.ReturnID>10);

