CSE 3241: Online Bookstore Project

# Overview

**Bits & Books** needs a simple information management system and database to support their inventory and sales operations.

The specification document and sample data will provide you with many of the user requirements.

In addition, to receive full credit for your project you will be required to come up with some extensions that expand on the requirements here and provide new functionality beyond the scope of the basic requirements.

# Worksheets, Checkpoints

You will start work on the project immediately and work on it incrementally all semester. Some work will be done during class time, but most of your effort will be with your partners.

Four checkpoint (draft) documents will be submitted during the semester so that your instructor can give you feedback and any necessary corrections.

You can find helpful worksheets and blank checkpoint documents on the class website.

# Final Project Document

The final report must be a professionally presented, well-organized, typed document, and a complete SQL database, submitted electronically to the Carmen Dropbox in a single ZIP file. This ZIP file needs to be neatly and professionally organized. With all filenames appropriately chosen, and all files suitably organized into subdirectories. Include a Table of Contents file named README.txt that explains the layout of your files, including where to find each of the following files in your file structure.

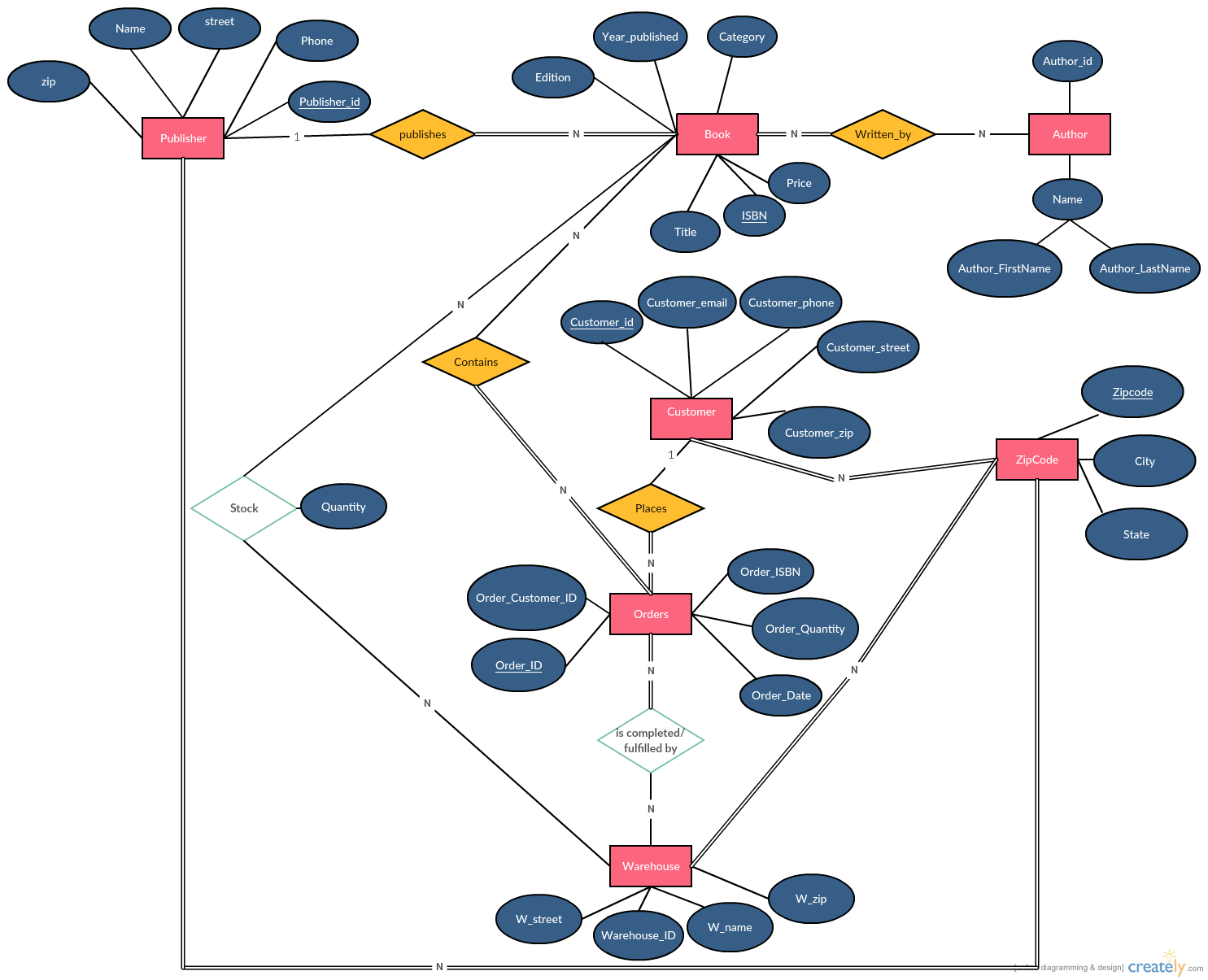
## Part I – The Final Report

Your final document will include a relational database schema, entity relationship diagram, SQL queries, and reports, as indicated below.

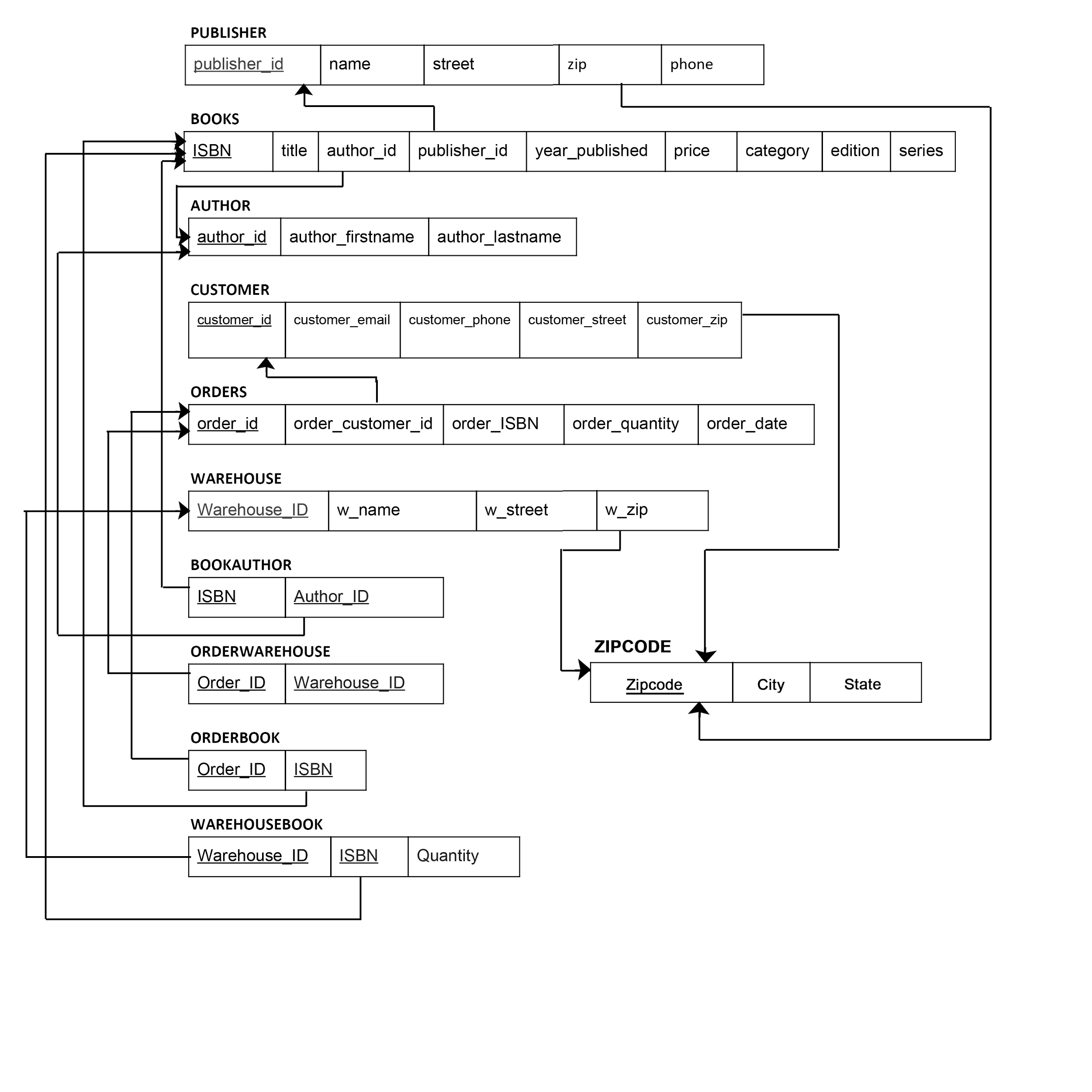
***Section 1 - Database Description***

A database description document that contains the following information about your database (compiled from your completed and revised checkpoints):

1. TYLER: A professionally presented, well-formatted ER-model that reflects the updates you have made during the semester. Do not submit a hand-drawn diagram. TYLER - UPDATE ER FROM LATEST CHECKPOINT. All attributes must match the relational schema. EX: for the Order entity, ISBN must be order\_ISBN. Delete SERIES on Books.



1. COURTNEY: A professionally presented, well-formatted relational schema for the database. This schema must be annotated with the primary key for each table, all foreign keys on all tables, and all functional dependencies on all tables. Make sure that connections between FKs and PKs are clear.



**Function Dependencies:**

Publisher: publisher\_ID → name, street, zip, phone

Book: ISBN → Title, Author\_id, Publisher\_id, Year\_published, Price, Category, Edition

Author: Author\_id → Author\_firstname, Author\_lastname

Customer: Customer\_id → Customer\_firstname, Customer\_lastname, Customer\_email, Customer\_phone, Customer\_street, Customer\_zip

Orders: Order\_id → Order\_customer\_id, Order\_ISBN, Order\_Date

Warehouse: Warehouse\_id → W\_name, W\_street, W\_zip

ZipCode: Zipcode → City, State

BookAuthor: ISBN → author\_id

OrderWarehouse: order\_id → warehouse\_id

OrderBook: {order\_id, ISBN} → order\_quantity

WarehouseBook: {warehouse\_id, ISBN} → quantity

1. COURTNEY: For each table, give a brief description of the level of normalization achieved for that table. If the table is not in BCNF, explain why.

Publisher is in BCNF because it is in 1NF, 2NF and 3NF, publisher\_ID → name, street, zip, phone and publisher\_id is a superkey.

Books is in BCNF because it is in 1NF, 2NF and 3NF, ISBN → Title, Author\_id, Publisher\_id, Year\_published, Price, Category, Edition and ISBN is a superkey.

Author is in BCNF because it is in 1NF, 2NF and 3NF, Author\_id → Author\_firstname, Author\_lastname and author\_id is a superkey.

Customer is in BCNF because it is in 1NF, 2NF and 3NF, Customer\_id → Customer\_firstname, Customer\_lastname, Customer\_email, Customer\_phone, Customer\_street, Customer\_zip and Customer\_id is a superkey.

Orders is in BCNF because it is in 1NF, 2NF and 3NF, Order\_id → Order\_customer\_id, Order\_ISBN, Order\_Date and Order\_id is a superkey.

Warehouse is in BCNF because it is in 1NF, 2NF and 3NF, Warehouse\_id → W\_name, W\_street, W\_zip and Warehouse\_id is a superkey.

ZipCode is in BCNF because it is in 1NF, 2NF and 3NF, Zipcode → City, State and Zipcode is a superkey.

BookAuthor is in BCNF because it is in 1NF, 2NF and 3NF, ISBN → author\_id where both ISBN and Author\_id are superkeys.

OrderWarehouse is in BCNF because it is in 1NF, 2NF and 3NF, order\_id → warehouse\_id where both warehouse\_id and order\_id are superkeys.

OrderBook is in BCNF because it is in 1NF, 2NF and 3NF, {order\_id, ISBN} → order\_quantity and {order\_id, ISBN} is the superkey.

WarehouseBook is in BCNF because it is in 1NF, 2NF and 3NF, {warehouse\_id, ISBN} → quantity and {warehouse\_id, ISBN} is the superkey.

1. A description of each of the indexes that you have chosen to implement on your database, along with rationale for each.

We have chosen three particular indexes on two different tables to better optimize the performance of our database.

1) In the BOOKS table, we will create a nonclustered index on author\_id. By indexing author\_id, queries relating authors to the books they’ve written will be optimized. For example, finding all the books and prices of those books for a certain author.

2) In the BOOKS table, we will also create a nonclustered index on publisher\_id. Similar to indexing on author\_id, this index will optimize all queries that relate publisher\_id to the information in the books table. For instance, if we were to query for all the books published by publisher X, this query would be optimized.

3) Lastly, in the ORDERS table, we will create a nonclustered index on customer\_id. Because we find that it will be useful for the bookstore to be able to understand customer information, by indexing on customer\_id, queries that involve finding the amount a customer has purchased, the books they’ve purchased and their quantities will all be optimized.

While these three indexes greatly improve read query performance, they do so at the cost of insert/delete query performance. Thus, we decided to keep these indexes based on two major assumptions. First, the BOOKS table will be used for read queries much more frequently than insert/delete queries. Because we have put two indexes in this table, the insert/delete query performance is worst; however, we believe the bookstore will need to fetch information about more often than adding or deleting books from their inventory. Next, our second assumption is that the insert/delete query performance in the ORDERS table is not worsened enough to convince us not to use that index. Because new orders will be frequently made, it’s important to understand that this index hurts insert performance, but does not hurt performance enough to convince us from using this index.

1. DARCY: For each view that you have implemented, provide the following:
   1. A brief description in English of what this view produces, and why it would be useful.
   2. Relational algebra expression to produce this view.
   3. SQL statements to produce the view.
   4. Sample output from the view, with 5-10 lines of data records shown.

The view Booksold is defined as a table that has book ISBNs and the number of copies sold. Booksold can be used to reference the popularity of books among customers and determine demand of each book.

CREATE VIEW BOOKSOLD AS

SELECT ISBN, count(\*) as number\_sold

FROM Books as B, Orders as O

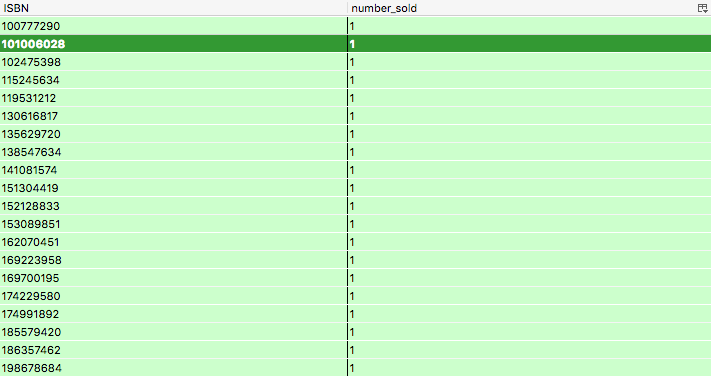
WHERE O.order\_ISBN = B.ISBN

GROUP BY ISBN;

RESULT ← ORDER⋈order\_ISBN = ISBN BOOKS

NUMBER\_SOLD ← Order\_IDζCOUNT ISBN (ORDERS)

BOOKS\_SOLD ← πISBN, NUMBER\_SOLD (RESULT)



The view Customerspend can be used to determine how many books a customer has bought and the amount of money the customer has spent at the store. This information can be valuable in creating customer loyalty programs and determining customer trends.

CREATE VIEW CUSTOMERSPEND AS

SELECT C.customer\_id, sum(OB.order\_quantity) as number\_books\_bought, sum(B.price \* OB.order\_quantity) as money\_spent

FROM Customer as C, Books as B, Orders as O, OrderBook as OB

WHERE C.customer\_ID = O.order\_customer\_id

AND OB.order\_id = o.order\_id

AND O.order\_ISBN = B.ISBN

GROUP BY C.customer\_id;

ALL\_CUSTOMERS ← CUSTOMER ⋈ customer\_ID = order\_customer\_ID ORDERS

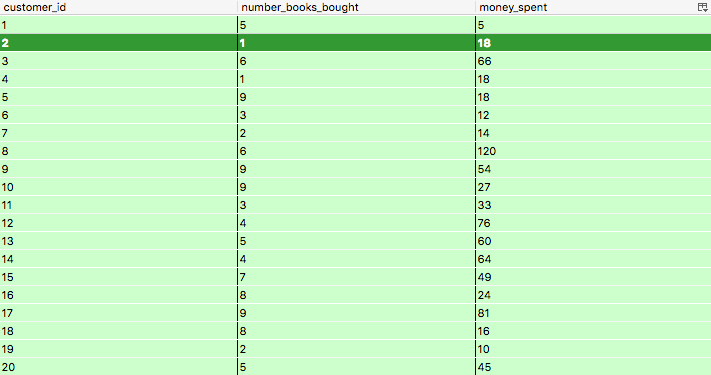
BOOK\_ORDERS ← ALL\_CUSTOMERS ⋈ order\_id = order\_id ORDERBOOK

ALL\_BOOKS ← BOOK\_ORDER ⋈order\_ISBN = ISBN BOOKS

NUMBER\_BOOKS\_BOUGHT ← customer\_id ζSUM (order\_quantity)

MONEY\_SPENT ← Customer\_id ζSUM(price \* order\_quantity)

Πcustomer\_id, NUMBER\_BOOK\_BOUGHT, MONEY\_SPENT ALL\_BOOKS



1. TYLER: A professionally presented description of three sample transactions useful for your database. This should include the sample SQL code for each transaction as well as an English language description of what “unit of work” the transaction represents. Remember – a transaction is a sequence of SQL statements taken as a unit – this can be reads and writes together or just a sequence of writes. One example of a sample transaction you might want to consider is the user making changes to an order – what might need to be considered a transaction in that case?

BEGIN TRANSACTION

Insert into ORDERS values (21, 20, 102475398, '10/20/2010');

Insert into ORDERBOOK

Values(21, 102475398, 3);

Insert into ORDERWAREHOUSE

VALUES (21, 21);

Update WAREHOUSEBOOK

SET quantity = quantity -3

WHERE ISBN = 20 AND warehouse\_id = 21;

COMMIT TRANSACTION;

This code will create a new order, with the order id = 21, customer id = 20, ISBN = 102475398 and a date. Then Inserts the new order into the Orderbook and Orderwarehouse tables. Finally, updates the quantity of the warehousebook table to subtract the amount that was ordered.

BEGIN TRANSACTION

UPDATE orderbook

SET order\_quantity = order\_quantity + 1

WHERE order\_id = 20;

Update WAREHOUSEBOOK

SET quantity = quantity - 1

WHERE warehouse\_id = 20;

COMMIT TRANSACTION;

This code will add an add 1 more of the same book to the order. By updating order\_quantity of the orderbook table and then updating the quantity of the warehouse.

BEGIN TRANSACTION

DELETE FROM AUTHOR

WHERE author\_id = 20;

DELETE FROM BOOKAUTHOR

WHERE author\_id = 20;

COMMIT TRANSACTION;

This code will delete an author from the database by deleting the row from the author table and then deleteing the row from the bookauthor table based on the author id.

***Section 2 - User Manual***

A user manual describing the usage of your database, for use by developers who are going to be writing code to use your database. Your manual should include:

1. For each table, explain what real world entity it represents. Provide a description of each attribute, including its data type and any constraints you have built-in.

The Books table is responsible for housing information regarding books available at the store. The Books table has seven attributes as described below:

* ISBN (int, PRIMARY KEY): uniquely identifies each book using an industry standard ISBN
* Title (varchar(255)): title of the book
* Author\_id (int, FORIEGN KEY): identifies the author’s unique attribute
* Publisher\_id (int, FORIEGN KEY): identifies the publisher’s unique attribute
* Year\_published (int(4)): year the book was published
* Price (DECIMAL(4,2)): price of the book
* Category (varchar(255)): category of the book
* Edition (int): edition of the book

The Publisher table is responsible for housing information regarding the publisher of a book. The Publisher table has five attributes as described below:

* Publisher\_id (int, PRIMARY KEY): uniquely identifies each publisher
* Name (varchar(50)): name of the publisher
* Street (varchar(50)): street address of the publisher
* Zip (int(5), FORIEGN KEY): zip code associated with the address of the publisher
* Phone (int(9)): phone number of the publisher

The Author table is responsible for housing information regarding the author of a book. The Author table has three attributes as described below:

* Author\_id (int, PRIMARY KEY): uniquely identifies each author
* Author\_firstname (varchar(50)): first name of the author
* Author\_lastname (varchar(50)): last name of the author

The Customer table is responsible for housing information regarding the customer who has placed an order. The Customer table has seven attributes as described below:

* Customer\_id (int, PRIMARY KEY): uniquely identifies each customer
* Customer\_firstname (varchar(255)): first name of the customer
* Customer\_lastname (varchar(255)): last name of the customer
* Customer\_email (varchar(255)): customer email address
* Customer\_phone (int(9)): customer phone number
* Customer\_street (varchar(50)): street address of the customer
* Customer\_zip (int(5), FORIEGN KEY): zip code associated with the address of the customer

The Orders table is responsible for housing information regarding the order that the customer has placed. The Orders table has three attributes as described below:

* Order\_id (int, PRIMARY KEY): uniquely identifies each order placed by the customer
* Order\_customer\_id (int, FORIEGN KEY): references the unique customer who placed the order
* Order\_ISBN (int, FORIEGN KEY): references the book ISBN that is ordered
* Order\_Date (date): date the order was placed

The Warehouse table is responsible for housing information regarding where the books are physically stored. The Warehouse table has four attributes as described below:

* Warehouse\_id (int, PRIMARY KEY): uniquely identifies the warehouse
* W\_name (varchar(200)): name of the warehouse
* W\_street (varchar(200)): street address of the warehouse
* W\_zip (int(5), FORIEGN KEY): zip code associated with the address of the warehouse

The ZipCode table is responsible for housing geographicical information of the warehouse, the customer and the publisher. The ZipCode table has three attributes as described below:

* Zipcode (int(5), PRIMARY KEY): the unique zipcode of the town
* City (varchar(255)): name of the city
* State (varchar(255)): non-abbreviated name of the state

The OrderBook table is responsible for relating the Order and Book entities with respect to quantity of ISBN ordered by the customer. The OrderBook table has three attributes as described below:

* Order\_id (int, PRIMARY KEY, FORIEGN KEY): references the order\_id in the orders table
* ISBN (int, PRIMARY KEY, FORIEGN KEY): references the ISBN in the book table
* Order\_quantity (int): quantity of ISBN ordered by the customer

The BookAuthor table is responsible for relating the ISBN of the book table to the author\_id of the author table. The BookAuthor table has two attributes as described below:

* ISBN (int, PRIMARY KEY, FORIEGN KEY): references the ISBN in the book table
* Author\_id (int, PRIMARY KEY, FORIEGN KEY): references the author\_id in the author table

The OrderWarehouse table is responsible for relating the order\_id of the Order table to the warehouse\_id in the Warehouse table. The OrderWarehouse table has two attributes as described below:

* Order\_id (int, PRIMARY KEY, FORIEGN KEY): references the order\_id in the order table
* Warehouse\_id (int, PRIMARY KEY, FORIEGN KEY): references the warehouse\_id in the warehouse table

The WarehouseBook table is responsible for relating the Warehouse and Book entities with respect to quantity of ISBN available in the warehouse. The WarehouseBook table has three attributes as described below:

* Warehouse\_id (int, PRIMARY KEY, FORIEGN KEY): references the warehouse\_id in the warehouse table
* ISBN (int, PRIMARY KEY, FORIEGN KEY): references the ISBN in the book table
* Quantity (int): quantity of ISBNs available in the warehouse

1. MADISON/DARCY: The sample SQL queries that you provided in Checkpoints 03 and 02. These queries should be organized and presented neatly and professionally. Each query should include:
   1. An English language description of what the query should be returning
   2. The correct relational algebra syntax of the query
   3. The equivalent SQL query
2. INSERT syntax for adding new books, publishers, authors and customers to your system. If there are dependencies in your system that require multiple records to be added to tables in a specific order to add one of these items, make sure you clearly indicate what those restrictions are.

***BOOKS***

INSERT into BOOKS (ISBN, title, author\_id, publisher\_id, year\_published, price, category, edition) values (‘int’, ’string’, ‘int’, ‘int’, ‘int’, ‘int’, ’string’, ’string’);

***PUBLISHER***

INSERT into PUBLISHER (id, name, street, zip, phone) values (‘int’, ‘string’, ‘string’, ‘int’, ‘int‘);

***AUTHOR***

INSERT into AUTHORS (author\_id, author\_firstname, author\_lastname) values (‘int’, 'string', 'string');

***CUSTOMER***

INSERT into CUSTOMERS (customer\_id, customer\_email, customer\_phone, customer\_street, customer\_zip) values (‘int’, 'string', 'int', ‘string’, 'int');

***Restrictions:***

Every book must have an author and publisher. Thus in order to add a book into the database, there must exist a publisher\_id and author\_id in the PUBLISHER and AUTHOR tables for that book entity being.

1. DELETE syntax for removing books, publishers, authors and customers from your system. Again, indicate any dependencies that exist on the order that the steps in your DELETE must take. In addition, provide an example set of DELETE statements for each entity in your database.

***BOOKS***

DELETE from BOOKS where <search\_condition>;

***PUBLISHERS***

DELETE from PUBLISHER where <search\_condition>;

***AUTHORS***

DELETE from AUTHOR where <search\_condition>;

***CUSTOMERS***

DELETE from CUSTOMER where <search\_condition>;

***Restrictions:***

Every book must have an author and publisher. Thus in order to delete a publisher or author entity from the PUBLISHER or AUTHOR tables, there may not be a book in the BOOKS table that has a publisher\_id or author\_id as the foreign key from the entity being removed.

***Section 3 - Graded Checkpoint Documents***

An appendix to the final report that MUST contain all of your original, graded checkpoint documents organized in a neat and professional manner. TYLER: For each checkpoint that required a revision you MUST include a revision for that checkpoint. This revision may be a pointer to where in the final database document the “fixed” version of the checkpoint resides (“See Section X Page Y for the new relational model diagram” for example).

## Part II – The SQL Database

A binary version of your database, suitable for opening using either the sqlite3 command line tool or the Firefox SQLite Admin tool.

**SQL CREATE**. A text file containing all of the scripts needed to create your database schema on an empty database. This file should be properly commented and should execute properly if pasted into an sqlite command prompt (or loaded from the command line tool). These scripts should include all indexes and views created on your database.

**DATA FILES**. A set of text files containing the data to be loaded into your database. These files, when used with the table creation scripts above, should be able to recreate your database from scratch if your binary file is corrupted or lost. Make sure you provide instructions on how to use these scripts and files in a separate text file.

**SQL QUERIES**. A text file containing all of the SQL queries used in your final report from Part I. All of these queries must be in a form where they can be run over your database through a simple cut and paste into the admin tool or the sqlite3 command line tool. In addition, make sure that these queries are completely commented so that it is clear where the query comes from in your final report writeup. → need to combine .txt files into one titled “SQL\_Queries.txt”

**SQL INSERT/ DELETE**. A text file containing all of the sample INSERT and DELETE statements provided in your user manual, suitable for pasting into a command prompt and testing the result on your database.

*Figure 1. Book Data in .xls format (available on Carmen)*

