Rona Habashy
101120994

COMP 3005: Database Management Systems **(Due: Dec. 9, 2022 (11:59 PM))**

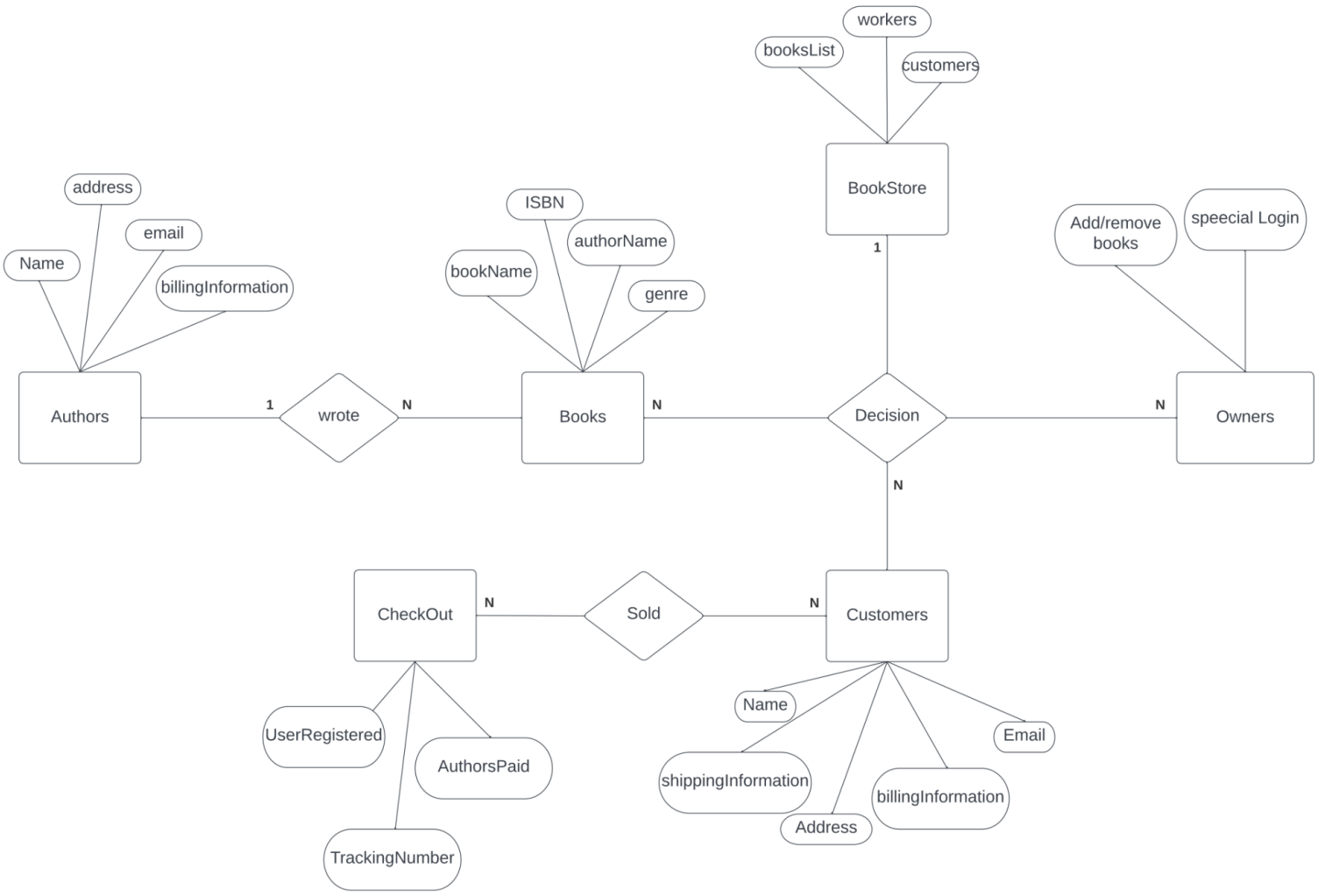
COMP 3005 Project (Fall 2022)

Instructor: Ahmed El-Roby and Abdelghny Orogat

1. This application lets users browse a collection of books that are available in the bookstore.

2. Project Report You need to submit one report file that contains the following sections. You can add other sections, but the following sections (except Bonus Features) must be in the report:

2.1. Conceptual Design (25%) This section should explain the conceptual design of the database. That is, the ER-diagram of the database for the bookstore and explanation of all the assumptions made in the diagram regarding cardinalities and participation types. Make sure that the assumptions do not contradict with the problem statement in Section 1.



2.1. Conceptual Design (25%)
Main entity BookStore.
Wrote relationship: 1:N
There is one author who could have written many different books.
Only one author could have written each book. (not including any co-author books)

Sold relationship: N:N
There are many customers, who can checkout as many times as they would like. There isn't a limit on the amount of times a customer or user is able to purchase from our bookstore.

Decision relationship: A three way relationship
BookStore - Books: 1:N:- there are many books being displayed and sold on the online bookstore.
BookStore - Customers: 1:N:- there are many customers or users, purchasing or just browsing our online bookstore.
BookStore - Owners: 1:N:- there could be more than one owner to our bookStore

2.2. Reduction to Relation Schemas (15%)

Reduce your ER-diagram into relation schemas and list these in this section.

```
CREATE TABLE author (
   nameID integer primary key not null,
   user_name varchar(20) NOT NULL,
   last_name text NOT NULL,
   year_born text NOT NULL,
   address varchar(20) NOT NULL,
   email varchar(20) NOT NULL,
   billingInformation varchar(20) NOT NULL
);
CREATE TABLE books (
   book_id integer NOT NULL primary key,
   publish_source text NOT NULL,
   publish_date  text NOT NULL,
   publish_author text NOT NULL,
   bookName varchar(20) NOT NULL,
   ISBN text NOT NULL,,
   genre varchar(20) NOT NULL,
   authorName varchar(20) NOT NULL,
   foreign key (authorName, nameID) references (nameID, author)

);
CREATE TABLE bookStore (
   bookList varchar(20) primary key not null,
   workers varchar(20) NOT NULL,
   customers varchar(20) NOT NULL
);
CREATE TABLE users (
   user_name varchar(20) primary key NOT NULL,
   first_name text NOT NULL,
   last_name text NOT NULL,
   year_born integer NOT NULL,
   shippingInformation varchar(20) NOT NULL,
   address varchar(20) NOT NULL,
   billingInformation varchar(20) NOT NULL,
   email varchar(20) NOT NULL,
   userID integer NOT NULL,
   foreign key (userID, userID) references (userID, checkOut)
);
```

```
CREATE TABLE checkOut (
   userID integer primary key not null,
   user_name varchar(20) NOT NULL,
   userRegistered text NOT NULL,
   authorsPaid text NOT NULL,
   trackingNumber integer,
   foreign key (user_name, user_name) references (user_name, users)
);

CREATE TABLE owners (
   specialLogin integer primary key not null,
   addBook varchar(20) NOT NULL,
   removeBook text NOT NULL
);

CREATE TABLE users (
 name VARCHAR(25) PRIMARY KEY,
 password VARCHAR(25) NOT NULL
);

INSERT INTO users (name, password) VALUES ("Rona", "password14");
INSERT INTO users (name, password) VALUES ("sandy", "password23");

CREATE TABLE books (
 title VARCHAR(255) PRIMARY KEY,
 author VARCHAR(255) NOT NULL,
 publisher VARCHAR(255) NOT NULL,
 price FLOAT NOT NULL,
 pages INTEGER NOT NULL,
 ISBN INTEGER NOT NULL
);

INSERT INTO books (title, author, publisher, price, pages, ISBN) VALUES ("The Cat in
the Hat", "Dr. Seuss", "Random House", 10.99, 10, 1890830336);
INSERT INTO books (title, author, publisher, price, pages, ISBN) VALUES ("To Kill a
Mockingbird", "Harper Lee", "HarperCollins", 11.99, 15, 9753150447);
INSERT INTO books (title, author, publisher, price, pages, ISBN) VALUES ("Dont Be
Sad", "Aid al-Qarni", "International Islamic Publishing House", 36.04, 330,
9780062457790);
INSERT INTO books (title, author, publisher, price, pages, ISBN) VALUES ("They Both
Die at the End", "Adam Silvera", "HarperTeen", 10.99, 384, 9860850447);
```

```
CREATE TABLE publishers (
 name VARCHAR(255) PRIMARY KEY,
 address VARCHAR(255) NOT NULL,
 phone VARCHAR(255) NOT NULL,
 bankAccount VARCHAR(255) NOT NULL
);

INSERT INTO publishers (name, address, phone, bank_account) VALUES ("Random House",
"New York, NY, USA", "212-555-1212", "1234-5678-9012-3456");
INSERT INTO publishers (name, address, phone, bank_account) VALUES ("HarperCollins",
"Toronto, ON, Canada", "416-555-1212", "9876-5432-1012-345");
INSERT INTO publishers (name, address, phone, bank_account) VALUES ("International
Islamic Publishing House", "Saudi Arabia", "416-587-6762", "9876-5432-1012-345");
INSERT INTO publishers (name, address, phone, bank_account) VALUES ("HarperTeen",
"Toronto, ON, Canada", "416-432-0975", "9876-5432-1012-345");
```

2.3. Normalization of Relation Schemas (20%)
Given the problem statement and your design, write the set of functional dependencies that apply to your database. Show that your relation schemas are either in a good normal form (show tests), or if they are not, show how to decompose them into a good normal form (show decomposition work), then show the **testing work** to make sure that they are in a good normal form.

```
author    --> nameID, user_name, last_name, year_born, address, email,
billingInformation
books     --> book_id, publish_source, publish_date, publish_author, bookName, ISBN,
genre, authorName
bookStore --> bookList, workers, customers
users     --> user_name, first_name, last_name, year_born, shippingInformation,
billingInformation, email, userID
checkOut  --> userID, user_name, userRegistered, authorsPaid, trackingNumber
owners    --> specialLogin, addBook, removeBook
```

The candidate key, we have one possible set of attributes forms candidate (minimal) key for a universal table consisting of all attributes with respect to the functional dependencies:

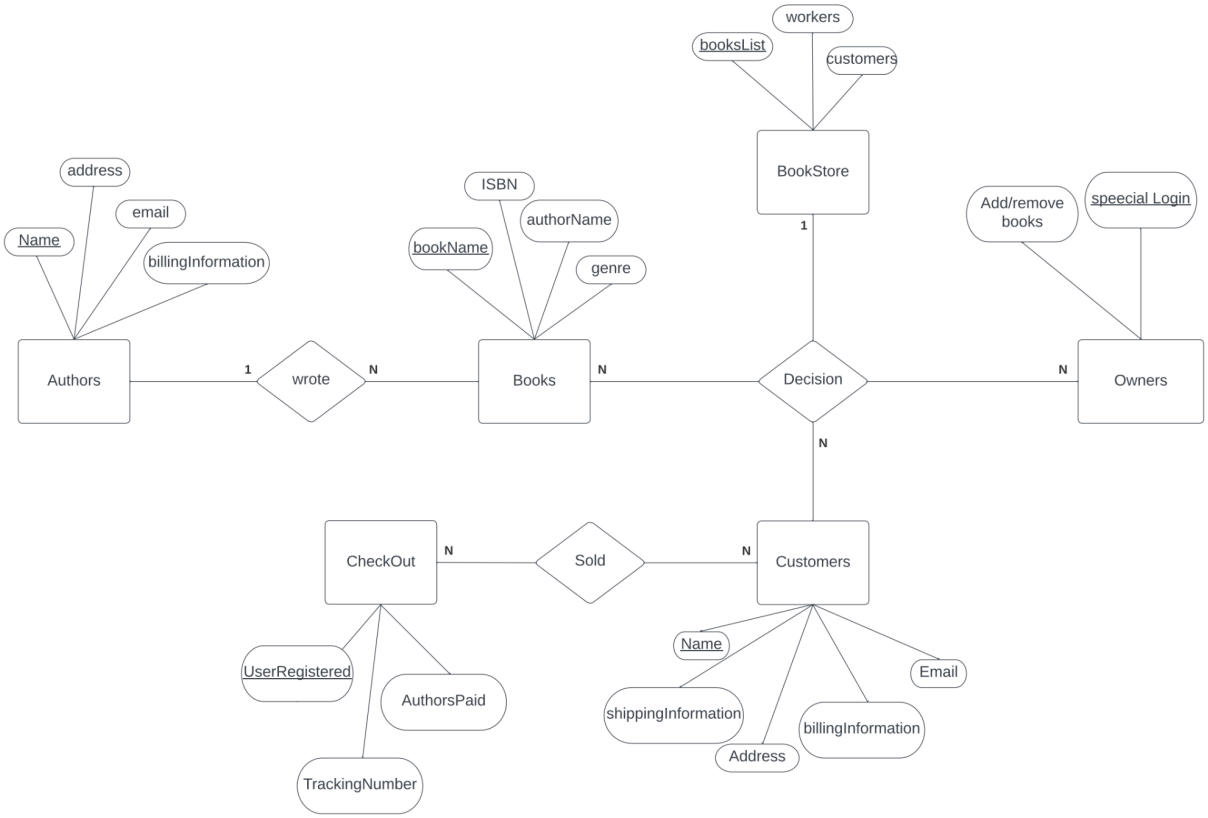
```
author, books, users, checkOut, and owners
```

Dependency Preserving, 3NF tables:
[Primary Key Attributes | Non-Primary Key Attributes]
[author | nameID, user_name, last_name, year_born, address, email, billingInformation]
[books | book_id, publish_source, publish_date, publish_author, bookName, ISBN, genre, authorName]
[bookStore | bookList, workers, customers]
[users | user_name, first_name, last_name, year_born, shippingInformation, billingInformation, email, userID]
[checkOut | userID, user_name, userRegistered, authorsPaid, trackingNumber]
[owners | specialLogin, addBook, removeBook]

Lossless join is a dependency preserving table, 3NF tables:
The definition of this collection of tables preserves all attributes and functional dependencies which can be joined without loss into a single relation consisting of all the attributes.

[author | nameID, user_name, last_name, year_born, address, email, billingInformation]
[books | book_id, publish_source, publish_date, publish_author, bookName, ISBN, genre, authorName]
[bookStore | bookList, workers, customers]
[users | user_name, first_name, last_name, year_born, shippingInformation, billingInformation, email, userID]
[checkOut | userID, user_name, userRegistered, authorsPaid, trackingNumber]
[owners | specialLogin, addBook, removeBook]
[author, books, users, checkOut, owners|]

2.4. Database Schema Diagram (10%)



2.5. Implementation (30%)

Description of the architecture of my application:
I am using the flask library with python to deploy my database, so that users/customers are allowed to interact with it.
Primary query demonstration: the books can be searched by the book name, author's name, ISBN, or by genre. When a customer or user selects a book, all necessary information is presented. Information such as the author's name, the genre the book is placed under, the publisher, the number of pages, the price of the book, and other similar books.
A basket has been added, where a user can add or remove as many books as they would like. The checkout basket can only be processed if the user is registered, with their billing information and shipping information saved.
Once an order has been placed, the customer is given a tracking number where they are able to enter it in our system to be able to track their order. Assuming all books are shipped from one warehouse, each order will receive one tracking number.
As you are on the login page, you are presented with two options, 1. Login as user or 2. Login as manager. Once a manager has logged in, they would have the ability to add or remove any book they like, they can also place orders for more stock of any of the books.
2.6. Bonus Features (Optional Section- Up to 15%)
Feature added is approximate search for books.

2.7. GitHub Repository All your source code for your application should be uploaded to a public GitHub repository1. The code needs to be well-documented and a decent README file that clearly states the instructions for running your code should be provided. The GitHub repository should also include a directory titled “SQL” that includes all the SQL DDL statements and SQL queries used in your application. You can organize the files in this directory in whichever way you would like. But all the files should have the .sql extension. For example, the directory “SQL” could have four files “DDL.sql”, “Functions.sql”, “Triggers.sql”, and “Queries.sql”. The files should be well-documented using comments in SQL (e.g., purpose of trigger, what a query retrieves from the database, etc.). These files are not meant to be executable/runnable. They are used for grading purposes. This section should include the url to this GitHub repository.