CS 377: Database Systems

Project #1 Solutions

SUBMISSION: Please submit the project electronically via Blackboard before 11:59 pm. Any submissions afterwards will be marked late and deducted accordingly unless you create a README file that specifies that you are using one of your 2 late assignment exceptions. You should use one file for each portion of the problem and follow these guidelines:

- If you have comments in your SQL query, use C style comments using /∗ < comment > ∗/
- Any text that is not within the comment brackets will be assumed to be part of your SQL command, and if the file does not execute properly, points will be deducted.
- Each file should be named as "<netID>-project1-<problem>-<part>.sql". For example, the instructor's answer to part b of question 2 would have the file "jho31-project1-2-b.sql".
- At the top of each of your submitted SQL files should be the honor code.

```
/* THIS CODE IS MY OWN WORK.
IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS.
_Your_Name_Here_ */
```

1. **Online Bookstore** (50 points): This problem is based in the Online Bookstore exercise from Homework #1. Your task is to design and implement the Relational Model of the Online Bookstore in MySQL. To perform this problem, you need to download and install MySQL from the following URL: `http://dev.mysql.com/downloads/mysql/`. The instructions for downloading and configuring MySQL are described in this video: `https://www.youtube.com/watch?v=iP1wOSsKjW8`.

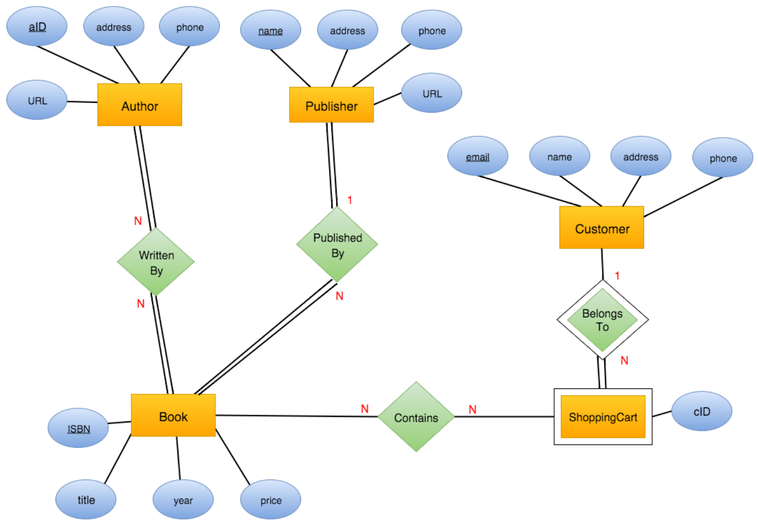

Figure 1: Online Bookstore Entity-Relation Model

(a) (15 points) Design the relational model for the online bookstore using Figure 1 as a reference. If you drew the relational model by hand, make sure to scan it and include it as a pdf or image.

(b) (15 points) Create the tables (implement the relations from the previous part) using MySQL. Write your commands in a SQL file named "<netID>-project1-1-b.sql" which follows the guidelines listed above.

(c) (10 points) Insert 3 records in each of the relations of the ER model. Put the commands in a file named "<netID>-project1-1-c.sql" which follows the guidelines listed above.

(d) (10 points) For each sold book (you can consider it sold if it is in the shopping cart), show the name of the customer, the book tittle, book name, and address of the publisher. Print your answer sorted by customer name.

(**ANSWER**)

(a) Figure 2 shows the relational model model.

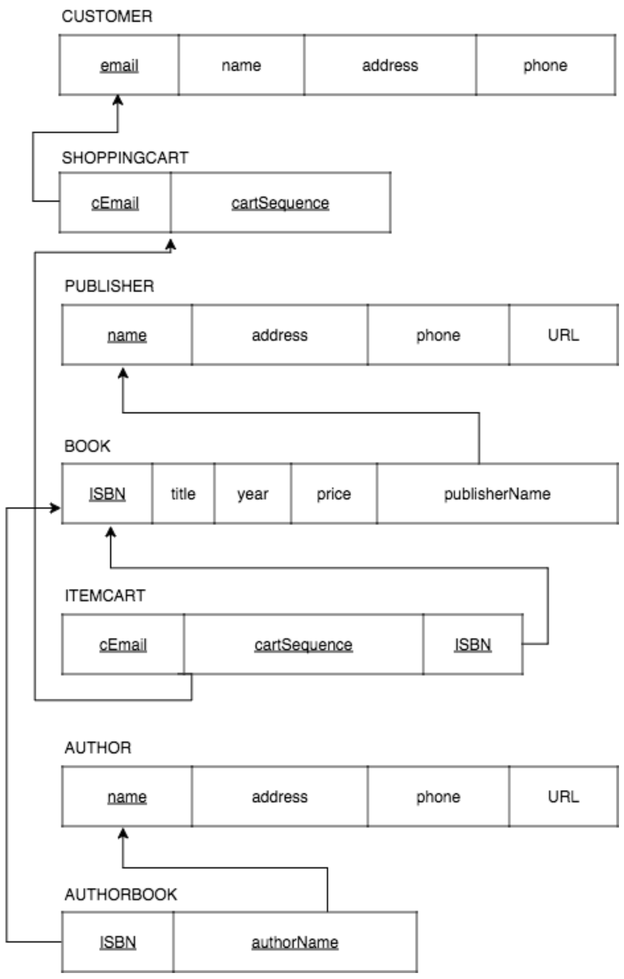

Figure 2: Online Bookstore Relational Model

(b) See project1-problem1.sql for the script to create the 7 relations in MySQL. The most important part of this exercise is to define all the primary keys and foreign keys in the proper order such that the tables will be created and you can insert tuples.

```
CREATE TABLE customer
(email VARCHAR(20) NOT NULL,
 name VARCHAR(40) NOT NULL,
 address VARCHAR(20) NOT NULL,
 phone INT NOT NULL,
 CONSTRAINT customerPK PRIMARY KEY (email));

CREATE TABLE shopping_cart
(cEmail VARCHAR(20) NOT NULL,
cartSequence INT NOT NULL,
CONSTRAINT cartPK PRIMARY KEY (cEmail,cartSequence),
CONSTRAINT cartCustomerFK FOREIGN KEY (cEmail) REFERENCES customer(email));

CREATE TABLE publisher
(name VARCHAR(30) NOT NULL,
address VARCHAR(20) NOT NULL,
phone INT NOT NULL,
URL VARCHAR(30) NOT NULL,
CONSTRAINT publisherPK PRIMARY KEY (name));

CREATE TABLE book
(ISBN INT NOT NULL,
title VARCHAR(30) NOT NULL,
year INT NOT NULL,
price DOUBLE NOT NULL,
publiName VARCHAR(30) NOT NULL,
CONSTRAINT bookPK PRIMARY KEY (ISBN),
CONSTRAINT bookPublisherFK FOREIGN KEY (publiName) REFERENCES publisher(name));

CREATE TABLE item_cart
(cEmail VARCHAR(20) NOT NULL,
cartSequence INT NOT NULL,
ISBN INT NOT NULL,
CONSTRAINT itemPK PRIMARY KEY (cEmail,cartSequence, ISBN),
CONSTRAINT itemCartFK FOREIGN KEY (cEmail, cartSequence) REFERENCES shopping_cart(cEmail, cartSequence),
CONSTRAINT itemBookFK FOREIGN KEY (ISBN) REFERENCES Book(ISBN));

CREATE TABLE author
(aID INT NOT NULL,
URL VARCHAR(30) NOT NULL,
address VARCHAR(30) NOT NULL,
phone INT NOT NULL,
CONSTRAINT authorPK PRIMARY KEY (aID));

CREATE TABLE author_book
(ISBN INT NOT NULL,
aID INT NOT NULL,
CONSTRAINT abPK PRIMARY KEY (ISBN,aID),
CONSTRAINT abBookFK FOREIGN KEY (ISBN) REFERENCES book(ISBN),
CONSTRAINT abAuthorPK FOREIGN KEY (aID) REFERENCES author(aID));
```

(c)
```
INSERT INTO customer
(email, name, address, phone)
VALUES
('cvalder@emory.edu',  'Camilo Valderrama', 'Street 1', 123 );

INSERT INTO customer
```

```
(email, name, address, phone)
VALUES
('joyce.c.ho@emory.edu', 'Joyce Ho', 'Street 2', 345);

INSERT INTO customer
(email, name, address, phone)
VALUES
('sam@emory.edu', 'Sam Simons', 'Street 3',  678 );

INSERT INTO publisher
(name, address, phone, URL)
VALUES
('Pearson', 'Street 4', 123, 'www.pearson.com' );

INSERT INTO publisher
(name, address, phone, URL)
VALUES
('Elsevier', 'Street 5', 123, 'www.elsevier.com' );

INSERT INTO publisher
(name, address, phone, URL)
VALUES
('McGraw-Hill', 'Street 6', 123, 'www.mcgrawhill.com' );

INSERT INTO book
(ISBN, title, year, price, publiName)
VALUES
(1234, 'Book 1', 2015, 30, 'McGraw-Hill');

INSERT INTO book
(ISBN, title, year, price, publiName)
VALUES
(4321, 'Book 2', 2016, 45, McGraw-Hill');

INSERT INTO book
(ISBN, title, year, price, publiName)
VALUES
(2589, 'Book 3', 2014, 32, 'Elsevier');

INSERT INTO shopping_cart
(cEmail, cartSequence)
VALUES
('cvalder@emory.edu', 1);

INSERT INTO shopping_cart
(cEmail, cartSequence)
VALUES
('joyce.c.ho@emory.edu', 1);

INSERT INTO shopping_cart
(cEmail, cartSequence)
VALUES
('joyce.c.ho@emory.edu', 2);

INSERT INTO item_cart
(cEmail, cartSequence, ISBN)
```

```
VALUES
('cvalder@emory.edu', 1, 1234);

INSERT INTO item_cart
(cEmail, cartSequence, ISBN)
VALUES
('joyce.c.ho@emory.edu', 1, 1234);

INSERT INTO item_cart
(cEmail, cartSequence, ISBN)
VALUES
('joyce.c.ho@emory.edu', 2, 4321);

INSERT INTO author
(aID, URL, address, phone)
VALUES
(789, 'www.a1.com', 'Street 28',  348);

INSERT INTO author
(aID, URL, address, phone)
VALUES
(456, 'www.a2.com', 'Street 98',  378);

INSERT INTO author
(aID, URL, address, phone)
VALUES
(654, 'www.a3.com', 'Street 94', 648);

INSERT INTO author_book
(ISBN, aID)
VALUES
(4321, 654);

INSERT INTO author_book
(ISBN, aID)
VALUES
(2589, 789);

INSERT INTO author_book
(ISBN, aID)
VALUES
(1234, 789);
```

(d)
```
SELECT c.name, b.title, p.name, p.address
FROM customer c, shopping_cart s, item_cart ic, book b, publisher p
WHERE c.email = s.cemail
AND ic.cEmail=s.cEmail
AND ic.cartSequence=s.cartSequence
AND ic.ISBN = b.ISBN
AND b.publiname = p.name
ORDER BY c.name;
```

2. **IMDB Database** (50 points): A copy of the IMDB database (`www.imdb.com`) has been loaded into the MySQL database on cs377spring.mathcs.emory.edu[1]. You can query the database either via the

[1]Not all the movies on the actual website will be reflected in this database. But the website will provide a fairly reasonable sanity check for your results.

MySQL client command line or a SQL GUI editor (e.g., MySQL Workbench) using the user 'cs377' with the password posted on Piazza. Detailed instructions for accessing the database via the command line will be provided on Piazza. Our IMDB database contains 6 relations which will be used for the second part of the project.

- actor(<u>id</u>, fname, lname, gender)
  - id = unique identifier for each actor
  - fname = first name of the actor
  - lname = last name of the actor
  - gender = gender of the actor

  This relation contains information on the actors that can be found in the IMDB database.
- movie(<u>id</u>, name, year)
  - id = unique identifier for each movie
  - name = name of the movie
  - year = year the movie was released

  This relation contains information on the movies that can be found in the IMDB database.
- director(<u>id</u>, fname, lname)
  - id = unique identifier for each director
  - fname = first name of the director
  - lname = last name of the director

  This relation contains information on the directors of the movies that can be found in the IMDB database.
- genre(mid, genre)
  - mid = foreign key referencing movie(id)
  - genre = the genre of the movie

  This relation contains information about the genre (classification) of the movies in IMDB.
- casts(aid, mid, role)
  - aid = foreign key referencing actor(id)
  - mid = foreign key referencing movie(id)
  - role = name of the character in the movie

  This relation contains information about the actors in each movie and the role they play.
- movie_director(did, mid)
  - did = foreign key referencing director(id)
  - mid = foreign key referencing movie(id)

  This relation contains information about the director(s) for each movie in IMDB.

Write SQL queries to answer the following questions. Each file should be named as "<netID>-project1-2-<part>.sql". For example, the instructor's answer to part b of this question would have the file "jho31-project1-2-b.sql".

(a) What genre(s) does the movie titled "Despicable Me" belong to?

(b) List all the animation movies that were released between 2011 and 2013 ordered by the year (2011 to 2013) and movie title (A to Z).

(c) Name the thriller movie(s) and the associated year of release directed by Steven Soderbergh sorted by the year (most recent first).

(d) List the movie(s) and the respective role(s) that Steve Carell was a part of as an actor.

(e) List the actors sorted by first name (A to Z) that have been directed by Woody Allen? Make sure to remove any duplicate names from the list.

(**ANSWER**) See project1-problem2.sql for the script to execute all the queries.

(a)
```
SELECT genre FROM movie, genre WHERE id = mid AND name = 'Despicable Me';
```

(b)
```
SELECT name, year FROM movie, genre
WHERE year <= 2013 AND year >= 2011
AND id = mid AND genre = 'Animation' order by year, name;
```

(c)
```
SELECT movie.name, year
FROM movie, genre, movie_director, director
WHERE movie.id = movie_director.mid AND movie_director.mid = genre.mid
AND did = director.id
AND fname = 'Steven' AND lname = 'Soderbergh'
AND genre = 'Thriller'
ORDER BY year DESC;
```

(d)
```
SELECT name, role
FROM movie, actor, casts
WHERE movie.id = mid AND actor.id = aid
AND fname = 'Steve' AND lname = 'Carell';
```

(e)
```
SELECT DISTINCT actor.fname, actor.lname
FROM actor, casts, director, movie_director
WHERE casts.mid = movie_director.mid and director.id = did
AND director.fname ='Woody' AND director.lname = 'Allen'
AND aid = actor.id
ORDER BY actor.fname ASC;
```