CS 377: Database Systems

Project #1 Solutions

SUBMISSION: Please submit the project electronically via Canvas before 11:59 pm. Any submissions afterwards will be marked late. You should use one file for each portion of the problem and follow these guidelines for your project submission:

- 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".
- A README.txt file **must** be submitted that contains the honor code, otherwise points will be deducted.

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

1. **Online Bookstore** (30 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. For instructions on installing MySQL, please see the Piazza post.

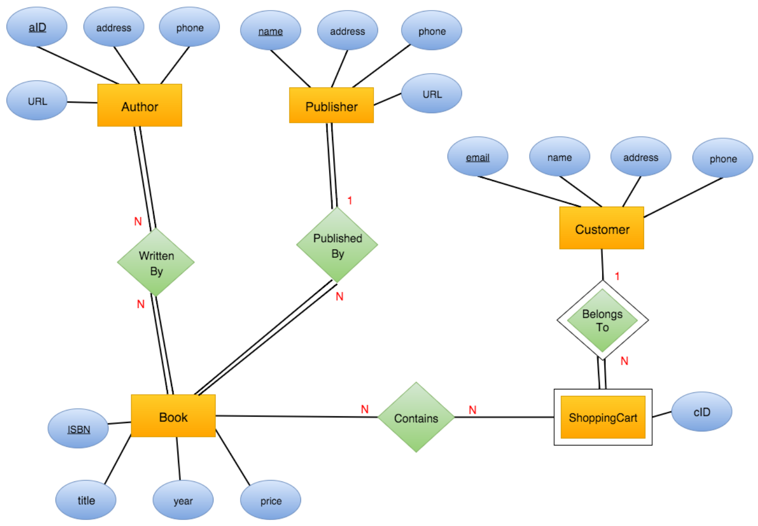

Figure 1: Online Bookstore Entity-Relation Model

(a) (10 points) Design the relational model for the online bookstore using Figure **??** as a reference. If you are drawing the relational model by hand, make sure to scan it and include it as a pdf or image.

(b) (10 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) (5 points) Insert 5 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) (5 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 **??** shows the relational model model.

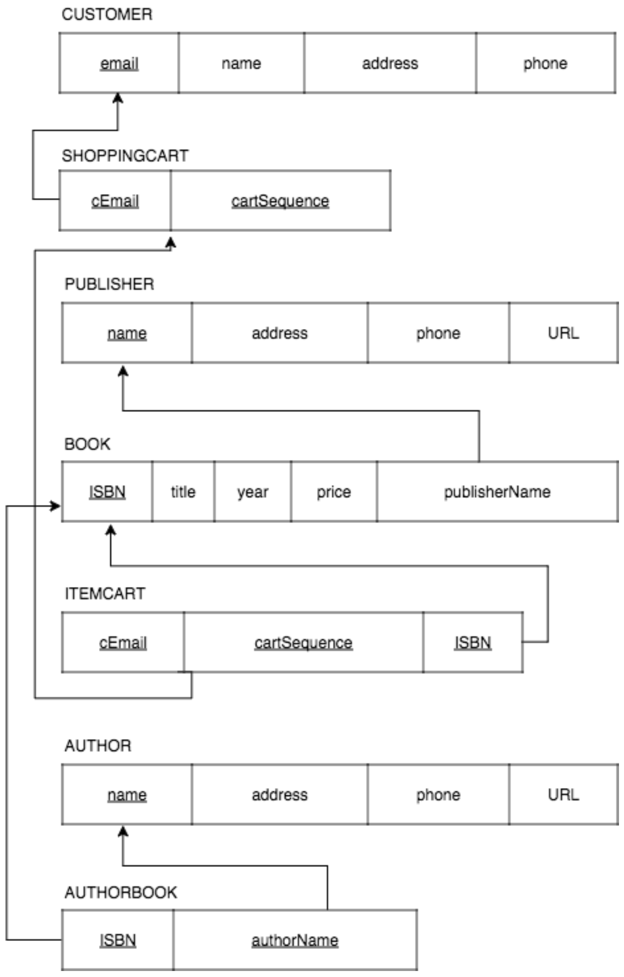

Figure 2: Online Bookstore Relational Model

(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** (70 points): A copy of the IMDB database[1] (`www.imdb.com`) has been loaded into the MySQL database on cs377db.mathcs.emory.edu. You can query the database either via the MySQL

[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.

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. In addition, we have provided the compressed database file for you to get a local copy, if you prefer to do the work locally. 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) (5 points) What genre(s) does the movie titled "Despicable Me" belong to?

(b) (5 points) 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) (5 points) Name the thriller movie(s) and the associated year of release directed by Steven Soderbergh sorted by the year (most recent first).

(d) (5 points) List the movie(s) and the respective role(s) that Steve Carell was a part of as an actor.

(e) (5 points) 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.

(f) (5 points) List all the movie names and director names where the director is also an actress (i.e., female) in the movie.

(g) (5 points) List the movies from 2004 where an actor plays more than one role.

(h) (5 points) How many unique actors are in each genre in 2010?

(i) (5 points) List all actors by their first and last name, sorted by the last name from A to Z, who have played in at least 10 different movies in 2004.

(j) (5 points) Who are the top 100 directors who have directed the most movies from 2005 to 2010, in descending order of the number of movies they have directed? Output their first name, last name, and number of movies directed.

(k) (5 points) List the actor's name, the movie name, and role for those who played an officer (e.g., police officer, officer [name], etc.) in 2011.

(l) (5 points) Find the directors with the last name Zapata who did not direct a movie in 2005.

(m) (10 points) List the name of the collaborators (actors in the same movie) and their associated number of movies for Meryl Strep.

(**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;
```

(f)
```
SELECT movie.name, actor.fname, actor.lname
FROM actor, casts, movie_director, director, movie
WHERE actor.fname = director.fname
AND actor.lname = director.lname
AND actor.gender = 'F'
AND casts.mid = movie_director.mid
AND director.id = movie_director.did
AND casts.aid = actor.id
AND movie.id = casts.mid;
```

(g)
```
SELECT DISTINCT name
FROM casts, movie
WHERE mid = id
AND year = 2004
GROUP BY aid, mid
HAVING COUNT(role) > 1;
```

(h)
```
SELECT genre, COUNT(DISTINCT aid)
FROM casts, genre, movie
WHERE genre.mid = casts.mid
AND movie.id = genre.mid
AND year = 2010
GROUP BY genre;
```

(i)
```
SELECT a.fname, a.lname, COUNT(mid)
FROM movie m, casts c, actor a
WHERE year = 2004 AND m.id = c.mid AND a.id = c.aid
GROUP BY aid
HAVING COUNT(mid) >= 10
ORDER BY a.lname;
```

(j)
```
SELECT d.fname, d.lname, COUNT(md.mid)
FROM director d, movie_director md, movie m
WHERE md.mid = m.id AND m.year >= 2005 AND m.year <= 2010 AND d.id = md.did
GROUP BY md.did
ORDER BY COUNT(md.mid) DESC
LIMIT 100;
```

(k)
```
SELECT fname, lname, name, role
FROM casts, movie, actor
WHERE mid = movie.id
AND actor.id = aid
AND year = 2011
AND role LIKE '%Officer%';
```

(l)
```
SELECT *
FROM director
WHERE id NOT IN
    (SELECT did
     FROM movie, movie_director
     WHERE year = 2005
     AND id = mid)
AND lname = 'Zapata';
```

(m)
```
SELECT fname, lname, id, COUNT(DISTINCT mid)
FROM actor, casts
WHERE mid IN
    (SELECT mid
     FROM actor, casts
     WHERE fname = 'Meryl'
     AND lname = 'Streep'
     AND aid = id)
AND aid <> (SELECT id
     FROM actor
     WHERE fname = 'Meryl'
     AND lname = 'Streep')
AND aid = id
GROUP BY id;
```