CS 377: Database Systems

Homework #1 Solutions

1. **Emory College University Database** (45 points): You were just hired to develop the Emory College Database. Design an ER diagram for the college based on the following requirements:

   (a) The Emory College consists of a number of Departments (e.g., MathCS, Physics, Psychology, etc.).

   (b) A department has a unique name, address, phone number, and can have a number of faculty members.

   (c) Each faculty member has a SSN, name, office, phone number, and salary.

   (d) Each faculty member belongs to a department.

   (e) A department has one department chair with a starting date.

   (f) A department always has a chair.

   (g) A department can teach a number of courses but may not teach any courses.

   (h) A course has a unique ID, a course name, and a description.

   (i) A course will not be offered by more than one department.

   (j) A semester has an ID (e.g., F14, S14, F15, etc.), a start date and an end date.

   (k) A number of sections of a course is offered in a semester and a section will be taught by one faculty member.

   (l) Not every course will be offered in a semester and some courses may have multiple offerings (multiple sections of the course).

   (m) A student has an ID, name, and address.

   (n) A student can enroll in one section of some course.

   (o) A student can be enrolled in multiple sections of *different courses* in one semester (enforcing different courses can be challenging to represent in the ER diagram, but students should be allowed to take multiple sections).

   Make sure your design represents all of the above information correctly. You must specify the **cardinality** and **participation constraints** in the relationships.

   Some frequently asked questions on this problem:

   - Does every student have to take at least one section? No.
   - Does every section have to have students enrolled? No.
   - Is at least one section taught in each semester? Yes.

- Does the entity section have any attributes not mentioned in the problem? No, but you can add some so long as you justify them.
- Does a faculty member have to teach at least 1 section? No.
- Does a course have to be offered in at least one section? No.
- Can a faculty teach more than one section each semester? Yes.
- Different sections in the same semester can be taught by different faculties and a section of the same course in different semesters can also be taught by different faculty members.

(**ANSWER**) The first step is to find the entities, or the "things" that are important enough to uniquely identify them with a number. From the description, we can discern the following entities:

- Department
- Faculty
- Course
- Semester
- Section: Note that if you don't make section as an entity, the design will be overly complicated. In our system, the university considers section an entity, when you register for a course you are actually registering for a section of a course. Each section in Emory is assigned a unique number.
- Student

The second step is to find the attributes for each entity. Attributes are properties that belong to a single entity, if you find a property that involves 2 (or more) entities, that property is not an attribute but a relationship. From the description, we get the following attributes for our entities:

- Department(<u>dName</u>, address, phone) - Requirement (b)
- Faculty(<u>SSN</u>, name, office, phone, salary) - Requirement (d)
- Course(<u>CourseID</u>, cname, description) - Requirement (h)
- Semester(<u>SemID</u>, startDate, endDate) - Requirement (j)
- Section(<u>SectionID</u>) - This one is less apparent and will be clear once we define the relationships
- Student(<u>StudentID</u>, name, address) - Requirement (m)

Final step is to find the relationships, which are the properties that involve 2 or more entities. We also need to find the properties of each relationship. Based on the description above, we can gather the following relationships and their properties:

- *Member*(Department, Faculty) - Requirement (b) "department has a number of faculty members"
  Relationship Properties:
  - 1 department has N faculty

  - 1 faculty belongs to 1 department
  - Department can have 0 faculty members (partial)
  - Faculty member must belong to some department (total)
- *Chair*(Department, Faculty) - Requirements (e) and (f)
  - 1 department has 1 chair faculty
  - 1 faculty chairs 1 department
  - Every department must have a chair (total). While the department can have 0 faculty members, empty departments can have some faculty from another department as a chair.
  - Not every faculty member is a chair (partial)
  - Each department chair has a starting date which is an attribute in the relation
- *Catalog*(Department, Course) - Requirement (g)
  - 1 department offers N courses
  - 1 course is offered by 1 department
  - A department need not offer any course (partial)
  - A course must be offered by some department (total)
- *Offer*(Section, Course) - Requirement (k) and (l)
  - 1 course can be offered in N different sections
  - 1 section belongs to exactly 1 course
  - Not every course will be offered as sections (partial)
  - A section must offer some course (total)
- *OfferedIn*(Section, Semester) - Requirement (k)
  - 1 semester can be offered in N sections
  - 1 section is offered in 1 semester
  - Not every semester will offer the section (partial)
  - A section must be offered in some semester (total)
- *Teaches*(Section, Faculty) - Requirement (k)
  - 1 section is taught by 1 faculty
  - 1 faculty can teach N sections
  - A section must be taught by some faculty (total)
  - Not all faculty have to teach some section (partial)
- *Takes*(Student, Section) - Requirement (n) and (o)
  - 1 student can enroll in N sections
  - 1 section can have N students enrolled
  - A student need not enroll in any section
  - A section can have no students enrolled (Note that sections without students will in general be cancelled but exist in the database before enrollment period)

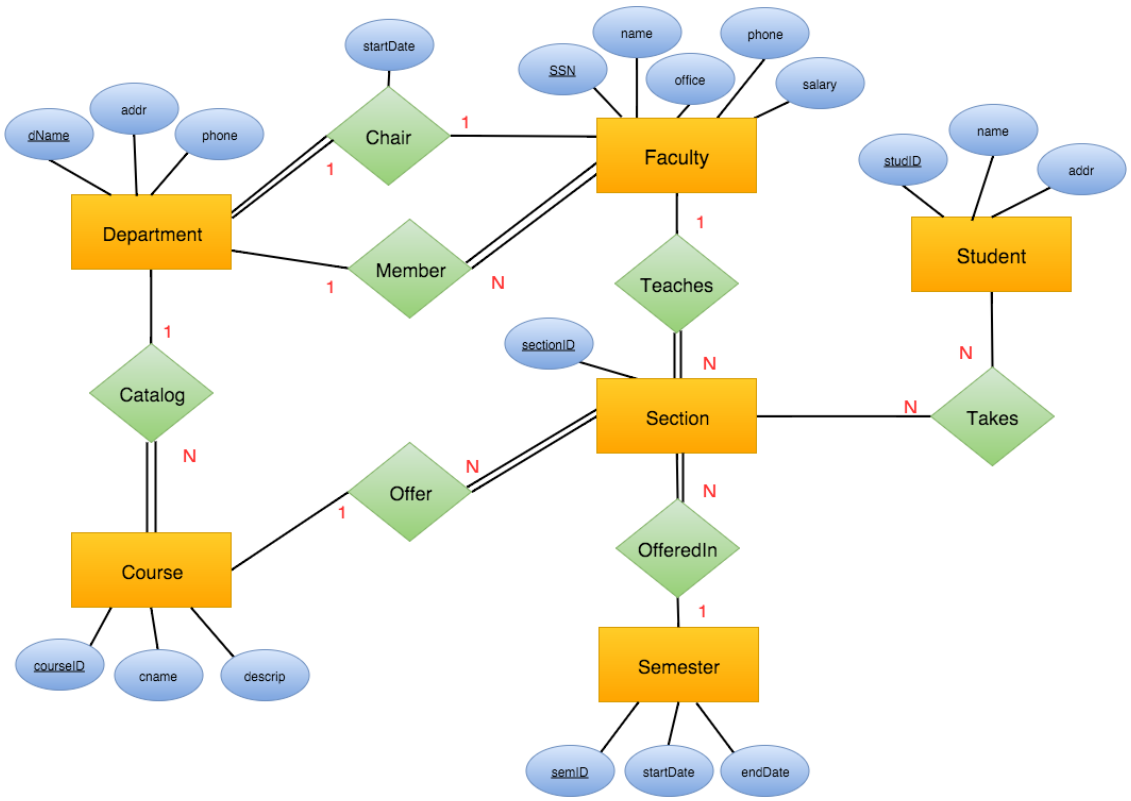

Figure 1: ER diagram for Emory College Database

The above information results in the diagram show in Figure 1. Note that section only has a single attribute, SectionID and that the constraint that students may not take different sections of the same course in the same semester cannot represented (easily) with an ER diagram.

2. **Online Bookstore** (25 points): You have been tasked with building an online bookstore, similar to Barnes & Noble (`www.barnesandnoble.com`). You may want to refer to the Barnes & Noble website to familiarize yourself. Your bookstore will be a simpler form with just the following requirements:

   (a) Each customer has a unique email, name, address, and phone number.

   (b) Customers can add books to their shopping cart (similar to adding items to your shopping cart in Amazon). Each customer can have multiple carts with different books. It is important to note that this is different than the traditional e-commerce websites which only allow each user to have 1 cart.

   (c) Each book has an ISBN, title, publication year, and the price.

   (d) A book is written by one or more authors.

   (e) Each author has a unique ID, name, address, and webpage URL.

   (f) Each book is published by a single publishing company (also known as a publisher).

   (g) A publisher has a unique name, address, phone, and webpage URL.

Create an ER diagram that represents the above information. You must specify the **cardinality** and **participation constraints** in the relationships. Note any unspecified requirements and your assumptions used to make the specification complete.

(**ANSWER**) The first step is to find the entities, or the "things" that are important enough to uniquely identify them with a number. From the description, we can discern the following entities:

- Customer
- Shopping Cart
- Book
- Author
- Publisher

The second step is to find the attributes for each entity. Attributes are properties that belong to a single entity, if you find a property that involves 2 (or more) entities, that property is not an attribute but a relationship. From the description, we get the following attributes for our entities:

- Customer(<u>email</u>, name, address, phone)
- ShoppingCart(cID) - For the solutions, shopping basket is a weak entity since it won't have much meaning without a customer. No points will be taken if you decide to make it a regular entity with the cartID as a primary key.
- Book(<u>ISBN</u>, title, year, price)
- Author(<u>aID</u>, name, address, URL)
- Publisher(<u>name</u>, address, phone, URL)

Final step is to find the relationships, which are the properties that involve 2 or more entities. We also need to find the properties of each relationship. Based on the description above, we can gather the following relationships and their properties:

- *BelongsTo*(Shopping Cart, Customer)
  Relationship Properties:
  - 1 customer has N Baskets
  - 1 basket belongs to 1 customer
  - Customer can have 0 baskets (partial)
  - Cart must belong to a customer (total). This is enforced since we have a weak entity and will be a weak relationship.
- *Contains*(Shopping Cart, Book)
  Relationship Properties:
  - A shopping cart can have N books
  - 1 book can belong to N shopping carts
  - A shopping cart can have 0 books (partial)
  - A book can belong to 0 shopping baskets (partial)

- *WrittenBy*(Book, Author)
  Relationship Properties:
  - A book can have N authors
  - An author can have N books
  - A book must have at least 1 author (total)
  - An author must have written at least 1 book (total).
- *PublishedBy*(Book, Publisher)
  Relationship Properties:
  - A book has 1 publisher
  - A publisher can publish N books
  - A book must have a publisher (total)
  - A publisher must have a book (total).

  Note that both PublishedBy and WrittenBy have total participation constraints on Authors and Publishers. Although your answers can choose to have a partial, this is mostly from a logistical perspective of not storing more information than is necessary.

The above information results in the diagram show in Figure 2.

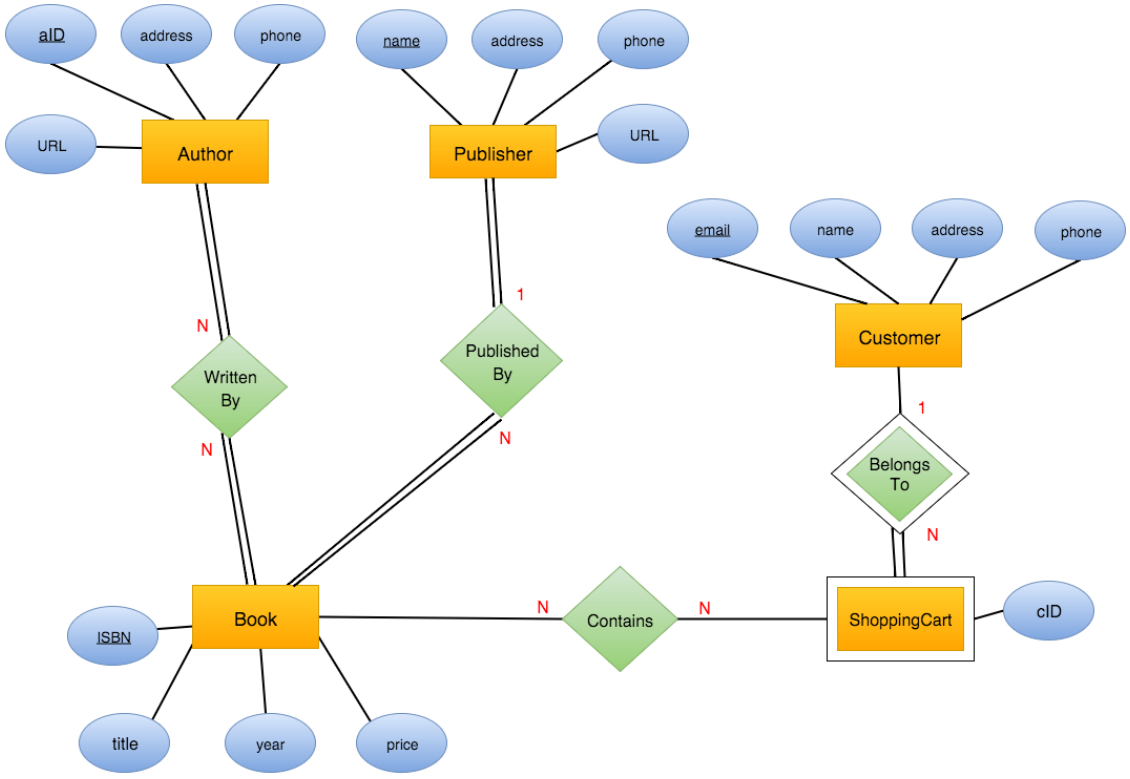

Figure 2: ER diagram for Online Bookstore

3. **Car Insurance Database** (30 points): Map the ER diagram for a car insurance company shown in Figure 3 into a relational database.

- Use the *smallest* possible number of relations
- Attributes in the resulting relations must not have NULL values
- Use the entity names in the ER-diagram as the name of the relation
- Use the attribute names in the ER-diagram for attribute names in the relational model
- If you augment an existing relation with an attribute x to represent a relationship R, give the attribute the name: $R.x$
- If you define a new relation to represent a relationship, use the name of the relationship for the name of the relation
- **Underline** the *primary key* in every relation
- Draw an arrow from each *foreign key* to its corresponding *primary key*

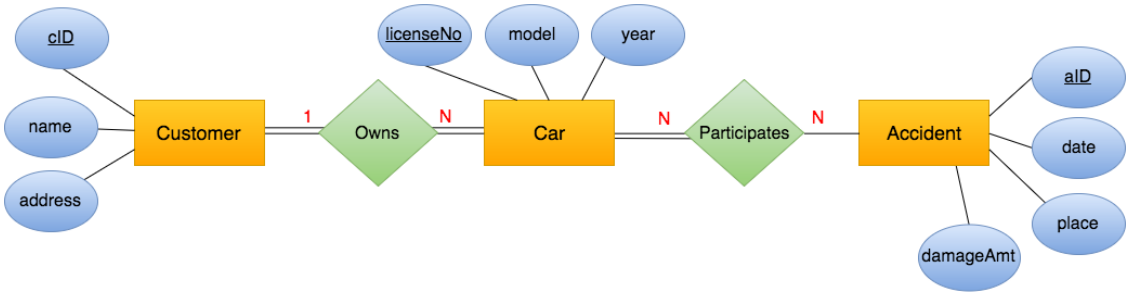

Figure 3: ER diagram for a car insurance company

(**ANSWER**) From the diagram, we first convert the entities into relations

- Customer(<u>cID</u>, name, address)
- Car(<u>licenseNo</u>, model, year)
- Accident(<u>aID</u>, date, place, damageAmt)

The next step is to take care of the relationships. We will favor expanding existing relations over creating new relations

- Owns(Customer, Car) introduces a new column, Owns.cID, in the Car relation that is a foreign key to the primary key in Customer because of the 1:N relationship
- Participates(Car, Accident) results in a new relation Participates(<u>aID</u>, <u>licenseNo</u>) because you can not easily expand an existing relation

Putting everything together yields the following relational data model (shown in Figure 4):

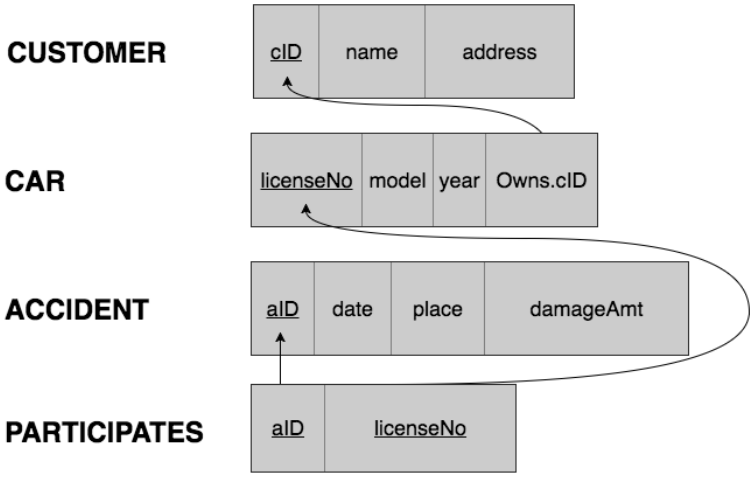

Figure 4: Relational Data Model for Car Insurance Company