CSE-3421: Exercises

---

1. **Independence**

   Answer #1.2 (page 23) from the textbook: What is *logical data independence* and why is it important?

   A short paragraph is sufficient.

2. **The House Database**

   You have just moved to Washington, D.C., to work for the U.S. House of Representatives (the "House") as a database specialist. For your first job, they want you to design a database to keep track of votes taken in the House. (A counterpart to you working for the U.S. Senate is designing a similar database for tracking votes in the Senate.)

   The database will track votes taken in the House during the current two-year congressional session. It should record each U.S. state (for instance, Texas) with its name, number of representatives, and *region* in which the state is located (northeast, mid-atlantic, midwest, and so forth). Each congress-creature, er, I mean representative, in the House is to be described by his or her name, the district (by district number) that he or she represents, the year when he or she was first elected, and the political party to which he or she belongs (for instance, *Republican*, *Democrat*, *Independent*, *Green*, *Reform*, *Other*). The database should track each bill (legislation) with its name, the date on which its vote was taken, whether the bill passed or failed (so the domain is *yes* and *no*), and its sponsors (the representatives who proposed the bill). The database should track how each representative voted on each bill (*yes*, *no*, *abstain*, *absent*).

   a. Design an entity-relationship (E-R) schema diagram for the above enterprise. Be careful to ensure that each of the attributes would be restricted to *legal* values (no pun...). State clearly any assumptions that you make. Also state any business rules—logic to which the database should adhere—that are not captured in your E-R diagram.

   b. Do you have any derived attributes in your diagram? Would these be problematic in the implemented database?

3. **Airplane, the SEQUEL** (no pun...)

   You have become a world-renowned specialist in airline database design. (Nothing like specializing, eh?) Scoundrel Airlines has come to you for help. Your mission, should you decide to accept it, is to design an E-R schema for the Airline's database for flight information and reservations. They provide you the following specifications.

   Scoundrel Airlines wants to keep information on its *airplanes*. They want to record the *airports* that they fly into and out of. Airport information should include the airport's code (such as 'RIC' for the Richmond-Williamsburg International (?) Airport), the name of the airport, the primary city it serves, and in which state or province it is. The database should keep track of the types of airplanes that the Airline owns, capturing the information of the number of seats, the manufacturer, and the model name.

   The database should model the fact that only certain types of airplane can land at certain airports. For instance, jumbo jets can land at RIC, but not at the Newport News-Williamsburg International (???) Airport (PHF). Every airplane is of a given airplane type.

   The database needs to keep track of flights. A *leg* of a trip is denoted by its departure information (from airport X, departure time) and its arrival information (to airport Y, arrival time). A particular airplane is assigned to a given flight leg on a given day. Each flight leg has a number of seats available to be reserved. Each *seat* on the airplane is *reserved* for a given customer, recording his or her name and telephone number.

   A *flight* is made up of a sequence of legs, for which they want to record the flight number, the flight fare, and whether the flight flies on weekdays or not.

   Design an E-R schema diagram to model Scoundrel Airline's database. Again, state clearly any assumptions that you make, and state any rules that are not captured in your E-R diagram.

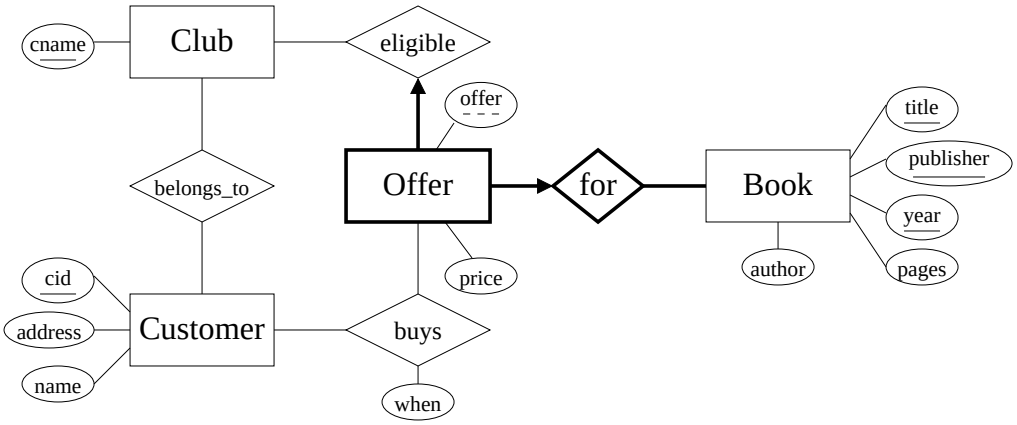

Figure 1: E-R diagram for Question 4.

4. **YRB.com**

You have just been hired at the famous company **YRB.com** (*York River Books*) which sells books over the Internet. They have been having problems with their database designs and have hired you as a database expert to help fix them.

Figure 1 shows the core of the E-R diagram that represents **YRB.com**'s main database to track sales of books to customers.

Customers belong to "clubs", such as *AAA*, *student*, *professor*, *AOL subscriber*, *AARP*, and so forth. There can be several "offers" available for a book which determines its price based upon the clubs to which a customer belongs.

a. How does one determine the price a given customer paid for a given book? Should an attribute price be added to **buys**?

b. Is it possible for two customers to buy the same book but for different prices? If so, how is this possible? If not, how does the logic of the E-R diagram prohibit this?

c. Does the customer always pay the lowest price for which he or she is eligible (**eligible**)? If not, is there an easy way to modify the E-R diagram in order to assure this?

d. Does the E-R diagram ensure that the Offer under which a customer buys a book is, in fact, legitimate? That is, an offer is for a particular club's members. Are we guaranteed that the customer belongs to the corresponding club? Why or why not?

5. **The Fix**

   Consider again **YRB.com**'s database (the E-R in Figure 1).

   a. There is a serious flaw (at least one) in the design in Figure 1, at least in as far as any bookseller would be concerned. What is the flaw?

   b. Redesign the E-R from Figure 1 to fix this.

Person(ssn, *name*, gender, *job*, studio, title, year)
FK (studio) refs **Studio**,
FK (title, year) refs **Movie**
Category(cat)
Movie(title, year, length, *director*, *studio*, *cat*)
FK (director) refs **Person** (ssn),
FK (studio) refs **Studio**,
FK (cat) refs **Category**
Cast(ssn, title, year, character)
FK (ssn) refs **Person**,
FK (title, year) refs **Movie**
Studio(studio, city)

Figure 2: Movies and Casts.

6. **Reverse Engineering**

A relational schema for movies and casts is given in Figure 2. The primary key for each relation is indicated by the underlined attributes, as ssn in **Person**. Attributes that are starred, as *director* in **Movie**, are not nullable (not null). (Likewise, any attribute which is part of the primary key is not nullable.) The foreign keys for each table are written as FK ..., and refs is shorthand for references.

The attributes title and year in table **Person** indicate that person's favorite movie. The attribute job of **Person** is allowed two values: 'actor' or 'director'. Only actors are allowed to act in (**Cast**) movies. Only directors are allowed to direct (director). A movie is produced by a studio. A person (actor or director) can be *contracted* by a studio (but by at most one studio), as indicated by studio in **Person**.

Reverse engineer this into a reasonable entity-relationship diagram that reflects the same logic of this relational schema and the explanation of the domain above.

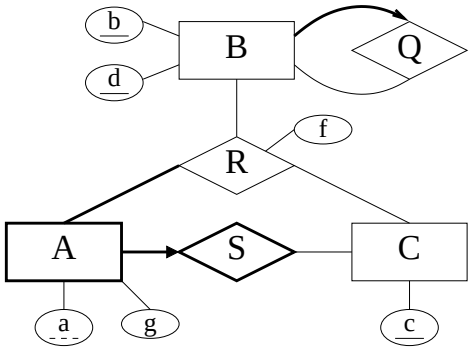

Figure 3: A, B, C's

7. **Scheming**

   Write a relational schema for the E-R in Figure 3. You may use the short-hand style like in Figure 2 from Question 6, rather than in full SQL CREATE TABLE statements. (Do this for all the components of the diagram—**A**, **B**, **C**, etc.—not just for part of it.) Be certain to capture the participation and key constraints from the E-R diagram in your relational schema, where possible. Do not introduce any tables that are not necessary. Note that **R** is a many-many-many.

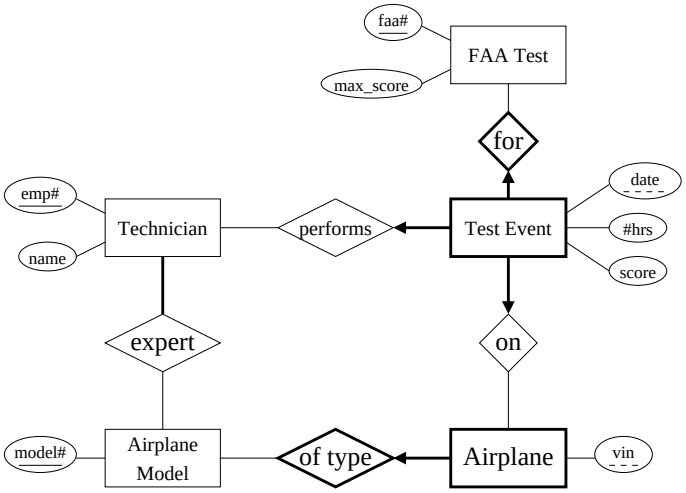

Figure 4: A, B, C's

8. **Extreme E-R**

   a. Redesign the E-R diagram from Figure 8 so that in the revised E-R diagram a technician who conducts a test on an airplane is guaranteed to be an expert on that model of airplane.

   b. Write a relational schema derived from your E-R design in 8a. Again, you may use the short-hand style as in Figure 2 from Question 6,

9. **Normal.** *Is this normal?*

   Consider the relation schema ABCDEF with the following set of functional dependencies:

$$\begin{array}{ll} A \mapsto B & AE \mapsto F \\ CD \mapsto A & CE \mapsto F \\ BC \mapsto D & \end{array}$$

   a. What are the keys of the relationship ABCDEF above with respect to the set of functional dependencies?

   b. Is schema ABCDEF in BCNF? Why or why not?

      Is schema ABCDEF in 3NF? Why or why not?

      Is schema ABCDEF in 2NF? Why or why not?

10. **Intersection by any other name...**

    Give two separate (*independent*) arguments (informal proofs) why the relational operator $\cap$ (*intersection* is algebraically redundant, given the relational operators σ (*selection*), π (*projection*), $\bowtie$ (*join*), $\cup$ (*union*), and $-$ (*set-difference*). That is, show that an R.A. query that uses intersection can always be rewritten as a query that does not.

11. **How many tuples for a tuppence?**

    Problem #4.2 from the textbook.

12. **What's that in English?!**

    Problem #4.4 from the textbook.

13. **One of these things is not like the other.**

Consider the schema $\mathbf{R}(\underline{\mathsf{A}}, \mathsf{B}, \mathsf{C})$, $\mathbf{S}(\underline{\mathsf{A}}, \underline{\mathsf{D}})$, and $\mathbf{T}(\underline{\mathsf{D}}, \mathsf{B}, \mathsf{E})$. Consider the following relational algebra expressions with respect to the above schema.

a. $\pi_{\mathsf{B}}((\sigma_{\mathsf{C}=5}(\mathbf{R}) \bowtie \sigma_{\mathsf{E}=7}(\mathbf{T})) \bowtie \mathbf{S})$

b. $\pi_{\mathsf{B}}(\pi_{\mathsf{A},\mathsf{B},\mathsf{D}}(\sigma_{(\mathsf{C}=5)\wedge(\mathsf{E}=7)}(\mathbf{R} \bowtie \mathbf{T})) \cap (\mathbf{S} \times \pi_{\mathsf{B}}(\mathbf{T})))$

c. $\pi_{\mathsf{B}}((\sigma_{\mathsf{C}=5}(\mathbf{R} \bowtie \mathbf{S})) \bowtie (\sigma_{\mathsf{E}=7}(\mathbf{S} \bowtie \mathbf{T})))$

d. $\pi_{\mathsf{B}}(\sigma_{\mathsf{C}=5}(\mathbf{R} \bowtie \mathbf{S})) \cap \pi_{\mathsf{B}}(\sigma_{\mathsf{E}=7}(\mathbf{S} \bowtie \mathbf{T}))$

e. $\pi_{\mathsf{B}}(\pi_{\mathsf{A},\mathsf{B}}(\sigma_{\mathsf{C}=5}(\mathbf{R})) \bowtie (\mathbf{S} \bowtie \pi_{\mathsf{D},\mathsf{B}}(\sigma_{\mathsf{E}=7}(\mathbf{T}))))$

One of these is not like the others. That is, one of them may evaluate differently from the other four. Which one?

Give an example relation for which this one does evaluate to a different answer than the other four, and show what the odd one evaluates to, and what the other four R.A. expressions evaluate to.

14. **Relational Algebra Queries.** *Parlez-vous l'algébra rélational?*

Customer	
cust#	PK
cname	
fav_colour	
phone#	

Item	
item#	PK
prod#	FK to **Product**
cust#	FK to **Customer**
colour	
date_sold	

Product	
prod#	PK
pname	
cost	
maker	FK to **Company**

Avail_Colours	
prod#	PK, FK to **Product**
colour	PK

Consider the relational schema above. PK stands for *primary key*, and FK for *foreign key*. Not all the schema is shown, but you do not need the parts not seen. Make natural assumptions about what the table attributes mean.

Write a relational algebra query for each of the following.

a. Show, for each customer (reporting the customer's name), the products by name that come in the customer's favourite colour.

b. Show, for each customer (reporting the customer's name), the products by name that *do not* come in the customer's favourite colour.

c. List pairs of customers (columns: first_cust#, first_cname, second_cust#, second_cname) such that the two customers own at least two products in common.

   Write your query so that a pair is not listed twice. For instance, if $\langle 5, bruce, 7, parke \rangle$ is listed, then $\langle 7, parke, 5, bruce \rangle$ should not be.

d. List customers who own items in all the available colours. That is, for every available colour, the customer owns some item in that colour.

e. List each customer by name, paired with the product(s) by name that he or she has bought that was the most expensive (cost) of all the products he or she has bought.

   Note that there actually may be ties. For instance, $\langle bruce, ferrari \rangle$ and $\langle bruce, porsche \rangle$ would both qualify if both were \$80,000, and for everything else he has bought, each item was less expensive than \$80,000.

15. **Domain Relational Calculus.** *Déjà vu to you too.*

Write the same queries as in Question 14, but as domain relational calculus queries instead.

a. Show, for each customer (reporting the customer's name), the products by name that come in the customer's favourite colour.

b. Show, for each customer (reporting the customer's name), the products by name that *do not* come in the customer's favourite colour.

c. List pairs of customers (columns: first_cust#, first_cname, second_cust#, second_cname) such that the two customers own at least two products in common.

Write your query so that a pair is not listed twice. For instance, if $\langle 5, bruce, 7, parke \rangle$ is listed, then $\langle 7, parke, 5, bruce \rangle$ should not be.

d. List customers who own items in all the available colours. That is, for every available colour, the customer owns some item in that colour.

e. List each customer by name, paired with the product(s) by name that he or she has bought that was the most expensive (cost) of all the products he or she has bought.

Note that there actually may be ties. For instance, $\langle bruce, ferrari \rangle$ and $\langle bruce, porsche \rangle$ would both qualify if both were \$80,000, and for everything else he has bought, each item was less expensive than \$80,000.

16. **Decomposition.** *Totally lossless!*

Consider the relation schema ABCDEF with the following set of functional dependencies:

$$\begin{array}{ll} \mathsf{A} \mapsto \mathsf{B} & \mathsf{AE} \mapsto \mathsf{F} \\ \mathsf{CD} \mapsto \mathsf{A} & \mathsf{CE} \mapsto \mathsf{F} \\ \mathsf{BC} \mapsto \mathsf{D} & \end{array}$$

Provide a lossless-join decomposition that is in BCNF for ABCDEF above with respect to the set of functional dependencies.

17. **Decomposition.** *Lose something?*

    Consider again the relation schema ABCDE, and the same set of functional dependencies, from Question 16. Dr. Dogfurry, the world famous database consultant, came up with the following decomposition: AEF, AB, and ACD.

    Is this a lossless join decomposition with respect to ABCDEF and the set of functional dependencies?

18. **Minimal Cover.** *Got you covered.*

    Given the set $\mathcal{F}$ of functional dependencies $\{\mathsf{AB} \mapsto \mathsf{C}, \mathsf{A} \mapsto \mathsf{D}, \mathsf{BD} \mapsto \mathsf{C}\}$, find a minimal cover of $\mathcal{F}$.