COSC-3421 Midterm Test w/ answers

- **Family Name:**
- **Given Name:**

- **Instructor:** Parke Godfrey
- **Exam Duration:** 75 minutes
- **Term:** Winter 2005

Your assignment, should you choose to accept it, is to answer the following questions to the best of your knowledge. Keep answers brief and to the point. Answers, such as *yes* or *no* should be justified by supporting argument, however. Be precise and be careful.

The exam is closed-book and closed-notes. Write any assumptions that you need to make along with your answers, whenever necessary.

There are five major questions. Each is worth ten points for a total of 50 points. The weight for each sub-question is indicated (or is one point by default).

Regrade Policy

- You must not write anything on the exam paper after the exam period if it is to be considered for regrading. Write any subsequent notes on separate paper.
- Regrading should be requested in writing. Write what you would like to be reconsidered. Note that an exam accepted for regrading will be reviewed and regraded in entirety (all questions).

Grading Box	
1.	
2.	
3.	
4.	
5.	
Total	

1. (10 points) **E-R.** *Where's the diagram?!* (Exercise)

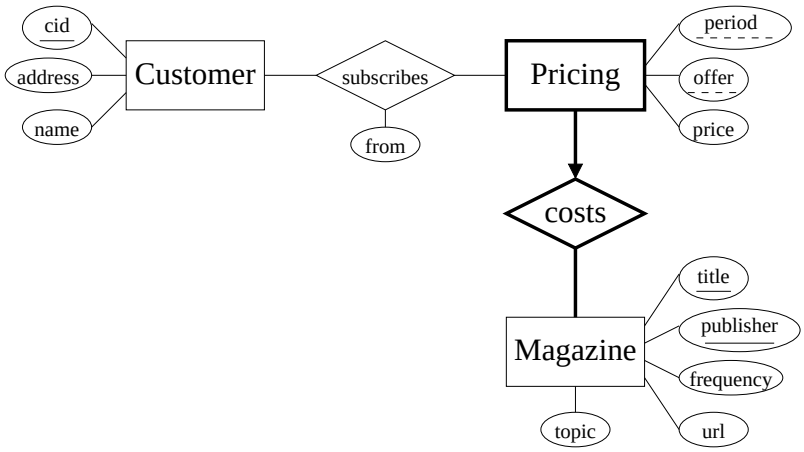
Customer(<u>cid</u>, *name*, *address*)
Magazine(<u>title</u>, <u>publisher</u>, *topic*, frequency, url)
Pricing(<u>title</u>, <u>publisher</u>, <u>period</u>, <u>offer</u>, *price*)
FK (title, publisher) refs **Magazine**
Subscribes(<u>cid</u>, <u>title</u>, <u>publisher</u>, <u>offer</u>, *from*, *period*)
FK (cid) refs **Customer**
FK (title, publisher, period, offer) refs **Pricing**

In each relation, the underlined attributes indicate the primary key. Attributes in italics (e.g., *name*) are not nullable. FK stands for *foreign key*. (These conventions are used throughout the test.)

The attribute period in **Pricing** tells for how many years the subscription is. The attribute frequency in **Magazine** tells how often the magazine is published (weekly, monthly, etc.). The attribute from in **Subscribes** tells when the subscription for the customer commences.

---

a. (7 points) Draw an E-R diagram for the relational schema above. Do not add any unnecessary elements. Do not make anything an entity that can be modeled appropriately as a relationship. Do not make anything a relationship (or an entity) that can be modeled appropriately as an attribute.



Note that this would mean that period is part of **subscribes**'s primary key. There is no *easy* way to model in E-R what we have exactly (that period is not part of **subscribes**'s key). So a further constrint we add to accompany the diagram is that a customer can subscribe to a magazine via a given offer once, so for just one period.

b. (3 points) Say that your schema additionally has

Offer(<u>offer</u>, <u>period</u>)

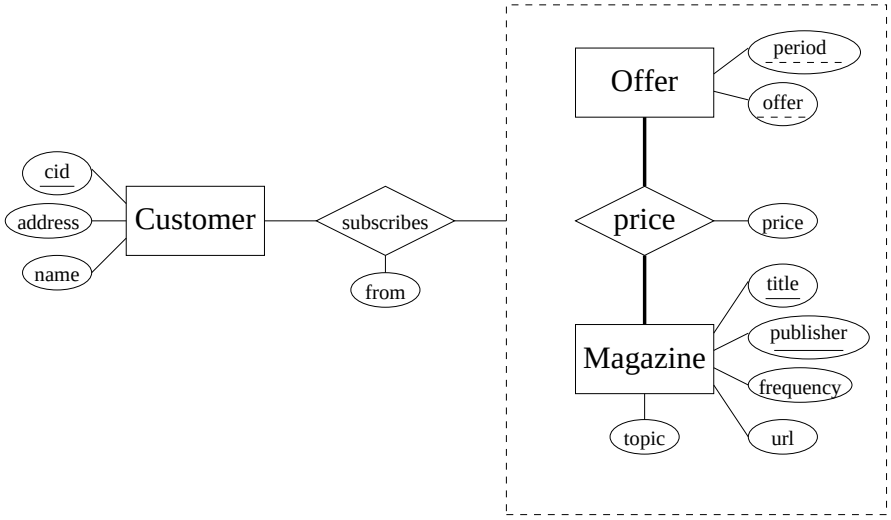
and **Pricing** is changed to

Pricing(<u>title</u>, <u>publisher</u>, <u>period</u>, <u>offer</u>, *price*)
FK (title, publisher) refs **Magazine**
FK (offer, period) refs **Offer**

(**Customer**, **Magazine**, and **Subscribes** stay the same as in 1a.)

Describe briefly in English how your E-R diagram (your answer in 1a) would need to be modified.

Now **Pricing** would be a relationship between **Offer** and **Magazine**. And **Subscribes** would be a relationship between **Customer** and the aggregate over **Pricing**.



2. (10 points) **The Relational Model / Miscellaneous.** *Table that motion!* (Multiple Choice)
   Pick the *one best* answer for each of the following.

---

a. Which of the following is *not* a relational schema? (The underlined attributes are to indicate the key.)

A. **marriage**(wife: name of the wife,
husband: name of the husband,
when: the date when they were married)

B. **personal**(name: name of the individual,
birthdate: when he or she was born,
from: where the individual lives,
hobbies: a list of that person's hobbies (references **hobby**))

C. **procedure**(name: a unique name for this piece of code,
language: which computer language it is written in (references **language**),
code: the code itself,
description: a description of what the code does)

D. **book**(title: the title of the book,
year: the year it was published,
author: who wrote it,
publisher: who published the book,
text: the entire text of the book)

E. **disease**(name: medical name of the disease,
symptom_1: boolean, whether the disease has symptom #1,
⋮
symptom_1219: boolean, whether the disease has symptom #1219)

---

b. A relational database management system (RDBMS) does *not* do which of the following?

A. Provide mechanisms to help protect the integrity of the data.

B. Ensure that relational schemas are in at least 3NF.

C. Allow for concurrent transactions against the database.

D. Facilitate crash recovery of the database in case of hardware failure.

E. Optimize query evaluation for arbitrary SQL queries.

---

c. Consider a relation with five attributes (say, A, B, C, D, and E). How many possibilities are there for what its primary key could be (e.g., ACE)?

A. 5 **B.** 25 **C.** 31 **D.** 120 **E.** ∞

---

d. Why are NULL values needed in the relational model? NULL values can be used for all but which one of the following?

A. To allow duplicate tuples in the table by filling the primary key column(s) with NULL.

B. To avoid confusion with actual legitimate data values like 0 for integer columns and ‘ ’ (the empty string) for string columns.

C. To leave columns in a tuple marked as “unknown” when the actual value is unknown.

D. To fill a column in tuple when that column does not really “exist” for that particular tuple.

E. To opt a tuple out of enforcement of a foreign key.

---

e. It is impossible to represent which of the following in a relational schema?

A. any mandatory participation constraint in a many-to-one relationship

B. any mandatory participation constraint in a many-to-many relationship

C. a one-to-one relationship

D. a many-to-one relationship

E. a ternary relationship

---

f. Schema normalization is *not* for

A. reducing the number of tables in the schema.

B. eliminating uncontrolled redundancy of data.

C. eliminating anomalies that could otherwise occur with *inserts*.

D. eliminating anomalies that could otherwise occur with *deletes*.

E. ensuring that functional dependencies are enforced.

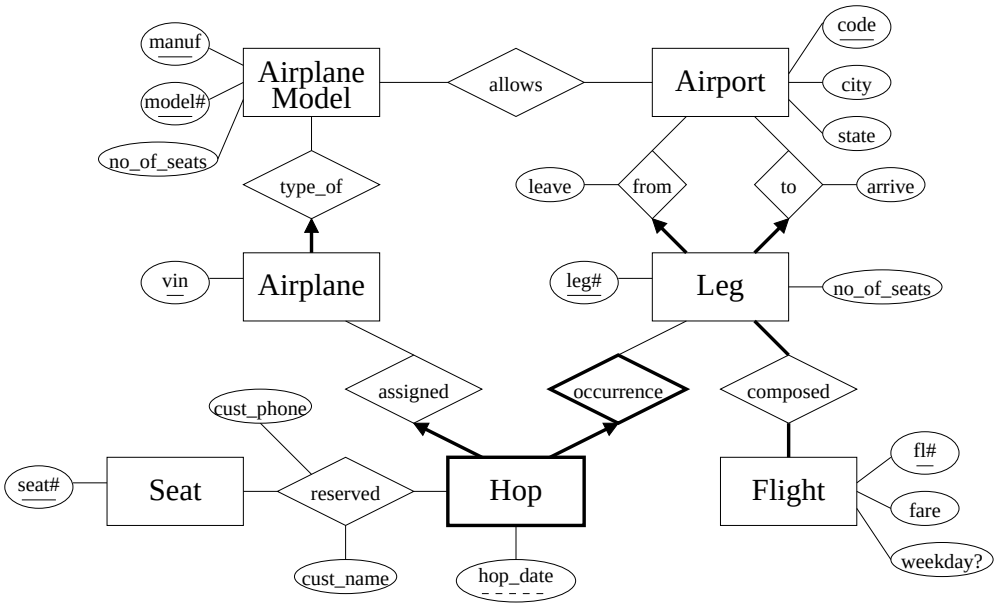


Figure 1: Flights.

Consider the E-R diagram in Figure 1 for the following questions.

---

g. How do we know the flight (fl#) to which a hop belongs?

A. We cannot know from the diagram as written because there is a mistake: **Hop** ought to have an attribute fl#.

B. By the logic of the E-R diagram, it does not make sense to ask which fl# to which a **Hop** belongs. **Hop** and **Flight** are not related.

C. A **Hop** is associated with an **Airplane**. An **Airplane** is associated with at most one **Flight**. Hence, we know the **Flight** to which a **Hop** "belongs" via its **Airplane**.

D. A **Hop** is associated with a **Leg** (via **occurrence**). We see fl# is the key of **Flight**. Hence, we know the **Flight** to which a **Hop** "belongs" via the **Leg** that they have in common.

E. By the logic of the E-R diagram, we do not know *the* **Flight**'s fl#, because a **Hop** can be associated with many flights.

---

h. How do we know what airport (code) a hop leaves *from*?

A. We cannot know from the diagram as written because there is a mistake: **Hop** ought to have an attribute from_code.

B. By the logic of the E-R diagram, it does not make sense to ask which **Airport** (code) a **Hop** departs *from*. **Hop** and **Airport** are not related.

C. A **Hop** is associated with **Airplane**. A given **Airplane** only leaves *from* one given **Airport**. Hence, we know the **Airport** *from* which a **Hop** departs via the **Airplane**.

D. A **Hop** is associated with **Leg** (via **occurrence**). A **Leg** leaves *from* a given **Airport**. Hence, we know which **Airport** the **Hop** leaves *from*.

E. By the logic of the E-R diagram, we do not know *the* **Airport**'s code a **Hop** departs *from*, because a **Hop** can be associated with many airports (via **from**).

---

i. Which of the following seems to be a logical problem with the E-R design (in Figure 1)?

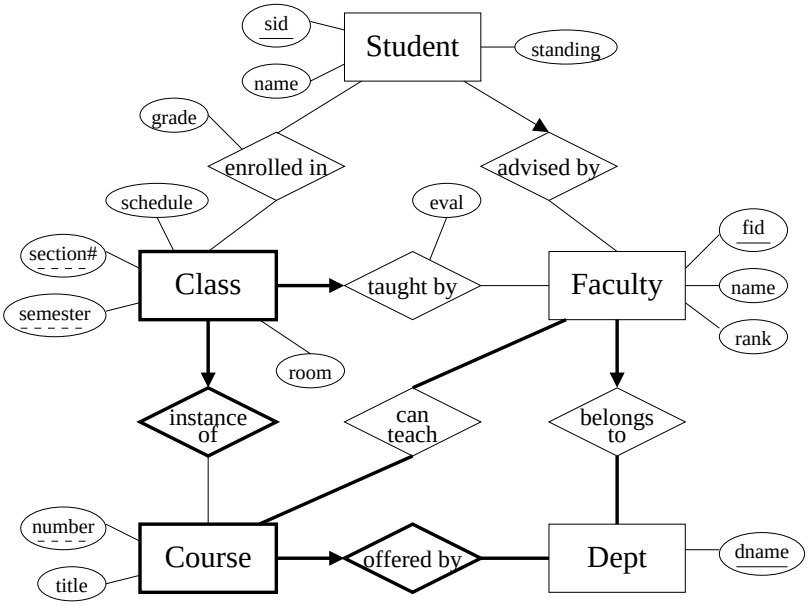
A. More than one customer may be assigned to a given **Seat** for a given **Hop**.
B. A given **Hop** may be assigned to more than one **Airplane**.
C. A given **Leg** may be assigned to more than one **Airplane**.
D. Different **Airplane_Model**s probably have different numbers of **Seat**s; the design does not capture this.
E. A **Flight** can be composed of at most two **Leg**s ("non-stop" flights).

---

j. Are we assured by the logic of the E-R diagram that the airplane assigned to a **Hop** (an occurrence of a scheduled **Leg**) is, in fact, allowed to take off from, and to land at, the airports for which the leg is scheduled to leave and arrive?

A. Clearly yes.
B. Clearly not.
C. It depends on one's interpretation of the diagram. Under one evident interpretation, the answer would be "yes". Under another, "no".
D. In the model, *all* airplanes are allowed at *all* airports.
E. Oddly, the diagram assures this for the departure airport, but not the arrival airport.

3. (10 points) **E-R to Schema.** *A Classy E-R.* (Exercise)



a. (7 points) Write a relational schema for the E-R diagram above. You may use the shorthand style used in exercises and in Question 1. Do not introduce tables unless they are completely necessary.

Dept(dname)
Faculty(fid, name, rank, *dname*)
FK (dname) refs **Dept**
Student(sid, name, standing, advisor)
FK (advisor) refs **Faculty** (fid)
Course(dname, number, title)
FK (dname) refs **Dept**
Class(dname, number, section#, semester, room, eval, *fid*)
FK (dname, number) refs **Course**
FK (fid) refs **Faculty**
can_teach(dname, number, fid)
FK (dname, number) refs **Course**
FK (fid) refs **Faculty**
enrolled_in(dname, number, section#, semester, sid, grade)
FK (dname, number, section#, semester) refs **Class**
FK (sid) refs **Student**

b. (3 points) How could you change your relational schema for Question 3a to ensure that courses offered by a department can be taught (that is, **can_teach**) only by faculty in that department?

You only need to describe briefly in English the changes. You do not need to show a new relational schema.

Change **Faculty**'s primary key to include dname. E-R-wise, this is making **Faculty** a weak entity on **Dept**. The table **can_teach** needs to have its FK to **Faculty** to now include dname. We do *not* change **can_teach** to have two attributes for "dname"! It still just has one, dname. So the only legal entry to **can_teach** is one for which the faculty member's department (dname) and the course's department (dname) are the same value.

4. (10 points) **Relational Algebra & Calculus.** *I rushed* $\Pi \bowtie \Sigma$. (Multiple Choice)
   Pick the *one best* answer for each of the following.

R	
A	B
1	2
3	2
5	6
7	8
9	8

S	
B	C
6	2
2	4
8	1
8	3
2	5

T	
A	C
7	1
1	2
9	3
5	4
3	5

Three tables: **R**, **S**, & **T**.

A	B	C
1	2	4
1	2	5
3	2	4
3	2	5
5	6	2
7	8	1
7	8	3
9	8	1
9	8	3

I

A	B	C
1	2	2
3	2	5
5	6	4
7	8	1
9	8	3

II

A	B	C
1	6	2
3	2	5
5	2	4
7	8	1
9	8	3

III

A	B	C
3	2	5
7	8	1
9	8	3

IV

A	B	C

V

Possible answer tables.

All the join operations below are natural joins.

a. What is the resulting table of $\mathbf{R} \bowtie (\mathbf{S} \bowtie \mathbf{T})$?
A. I **B. II** **C. III** **[D.] IV** **E. V**

b. What is the resulting table of $(\mathbf{S} \bowtie \mathbf{R}) \bowtie \mathbf{T}$?
A. I **B. II** **C. III** **[D.] IV** **E. V**

c. What is the resulting table of $\pi_{\mathsf{A,B}}(\mathbf{R} \bowtie \mathbf{S}) \bowtie \pi_{\mathsf{A,C}}(\mathbf{S} \bowtie \mathbf{T})$?
A. I **[B.] II** **C. III** **D. IV** **E. V**

d. What is the resulting table of $\pi_{\mathsf{A,B}}(\mathbf{R} \bowtie \mathbf{T}) \bowtie \pi_{\mathsf{B,C}}(\mathbf{S} \bowtie \mathbf{T})$?
[A.] I **B. II** **C. III** **D. IV** **E. V**

---

e. Let table $\mathbf{R}$ have attributes A and B, and table $\mathbf{S}$ likewise have attributes A and B. The relational algebra expressions below are equivalent except for which one?

A. $\mathbf{R} \cap \mathbf{S}$
B. $\mathbf{R} - (\mathbf{S} - \mathbf{R})$
C. $\mathbf{S} - (\mathbf{S} - \mathbf{R})$
D. $\mathbf{R} \bowtie \mathbf{S}$
E. $(\mathbf{R} \cup \mathbf{S}) - ((\mathbf{R} - \mathbf{S}) \cup (\mathbf{S} - \mathbf{R}))$

---

f. Consider the schema

$\mathbf{R}(\underline{\mathsf{A}}, B)$ FK (B) refs $\mathbf{S}$
$\mathbf{S}(A, \underline{\mathsf{B}})$ FK (A) refs $\mathbf{R}$

(None of the attributes are nullable.)

Which of the following is guaranteed to produce as many as, or more, tuples than each of the others?

A. $\mathbf{R} \bowtie \mathbf{S}$
B. $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \mathbf{S}$
C. $\mathbf{R} \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
D. $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
E. There is not enough information to answer this.

---

g. Consider the schema from Question 4f again.

Which of the following is guaranteed to produce fewer than, or at most the same, number of tuples as any of the others?

A. $\mathbf{R} \bowtie \mathbf{S}$
B. $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \mathbf{S}$
C. $\mathbf{R} \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
D. $\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \pi_{\mathsf{B}}(\mathbf{S})$
E. There is not enough information to answer this.

---

h. Consider the schema $\mathbf{R}(\underline{\mathsf{A}}, \underline{\mathsf{B}})$ and $\mathbf{S}(\underline{\mathsf{B}}, \underline{\mathsf{C}})$ (and no FKs).

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

A. $(\mathbf{R} - \mathbf{S}) \bowtie (\mathbf{S} - \mathbf{R})$
B. $(\mathbf{R} \bowtie \mathbf{S}) - \sigma_{\mathsf{B}=\mathsf{C}}(\mathbf{R} \bowtie \mathbf{S})$
C. $\sigma_{\mathsf{B}\neq\mathsf{C}}(\mathbf{R} \bowtie \mathbf{S})$
D. $(\pi_{\mathsf{A}}(\mathbf{R}) \bowtie \mathbf{S}) - \mathbf{R}$
E. $(\mathbf{S} - \mathbf{R}) \bowtie (\mathbf{R} - \mathbf{S})$

---

i. Consider the relations $\mathbf{R}(\mathsf{A}, \mathsf{B})$ and $\mathbf{S}(\mathsf{B}, \mathsf{C})$, and the following queries over $\mathbf{R}$ and $\mathbf{S}$.

I. $\{\langle \mathsf{A},\ \mathsf{C}\rangle \mid \exists \mathsf{B}(\langle \mathsf{A},\ \mathsf{B}\rangle \in \mathbf{R} \wedge \langle \mathsf{B},\ \mathsf{C}\rangle \in \mathbf{S})\}$

II. select distinct A, C from **R**, **S** where **R**.B = **S**.B

III. $\pi_{\mathsf{A},\mathsf{C}}(\mathbf{R} \bowtie \mathbf{S})$

One can determine that:

A. **I** & **II** are the same query.
B. **I** & **III** are the same query.
C. **II** & **III** are the same query.
D. **I**, **II**, & **III** are the same query.
E. **I**, **II**, & **III** are all different queries.

---

j. Consider the schema

R(<u>A</u>, *B*) FK (B) refs **S**
S(<u>B</u>, *C*) FK (C) refs **T**
T(<u>C</u>, *A*) FK (A) refs **R**

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

A. $\{\langle \mathsf{A}\rangle \mid \exists\, \mathsf{B}\ (\langle \mathsf{A},\ \mathsf{B}\rangle \in \mathbf{R} \wedge \exists\, \mathsf{C}\ (\langle \mathsf{B},\ \mathsf{C}\rangle \in \mathbf{S} \wedge \langle \mathsf{C},\ \mathsf{A}\rangle \in \mathbf{T}))\}$
B. $\{\langle \mathsf{A}\rangle \mid \exists\, \mathsf{B}\ (\langle \mathsf{A},\ \mathsf{B}\rangle \in \mathbf{R} \wedge \forall\, \mathsf{C}\ (\langle \mathsf{B},\ \mathsf{C}\rangle \in \mathbf{S} \rightarrow \langle \mathsf{C},\ \mathsf{A}\rangle \in \mathbf{T}))\}$
C. $\{\langle \mathsf{A}\rangle \mid \forall\, \mathsf{B}\ (\langle \mathsf{A},\ \mathsf{B}\rangle \in \mathbf{R} \rightarrow \forall\, \mathsf{C}\ (\langle \mathsf{B},\ \mathsf{C}\rangle \in \mathbf{S} \rightarrow \langle \mathsf{C},\ \mathsf{A}\rangle \in \mathbf{T}))\}$
D. $\{\langle \mathsf{A}\rangle \mid \exists\, \mathsf{C}\ (\langle \mathsf{C},\ \mathsf{A}\rangle \in \mathbf{T} \wedge \exists\, \mathsf{B}\ (\langle \mathsf{B},\ \mathsf{C}\rangle \in \mathbf{S} \wedge \langle \mathsf{A},\ \mathsf{B}\rangle \in \mathbf{R}))\}$
E. $\{\langle \mathsf{A}\rangle \mid \exists\, \mathsf{C}\ (\langle \mathsf{C},\ \mathsf{A}\rangle \in \mathbf{T} \wedge \forall\, \mathsf{B}\ (\langle \mathsf{B},\ \mathsf{C}\rangle \in \mathbf{S} \rightarrow \langle \mathsf{A},\ \mathsf{B}\rangle \in \mathbf{R}))\}$

1NF: Domain of each attribute is an *elementary* type; that is, not a *set* or a *record structure*.

2NF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is not a proper subset of any key for **R**.

3NF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then either

- A is *prime*, or
- $\mathcal{X}$ is a key or a super-key for **R**.

BCNF: Whenever $\mathcal{X} \mapsto A$ is a functional dependency that holds in relation **R** and $A \notin \mathcal{X}$, then

- $\mathcal{X}$ is a key or a super-key for **R**.

An attribute A is called *prime* if A is in any of the candidate keys.

---

5. (10 points) **Schema Refinement.** *My schema is fine enough, thank you very much.* (exercise)

   Consider the relation **R** with attributes A, B, C, D, E, F, and G and with the following functional dependencies (FDs):

   $$\mathsf{AD} \mapsto \mathsf{E} \qquad \mathsf{BE} \mapsto \mathsf{F}$$
   $$\mathsf{B} \mapsto \mathsf{C} \qquad \mathsf{AF} \mapsto \mathsf{G}$$

---

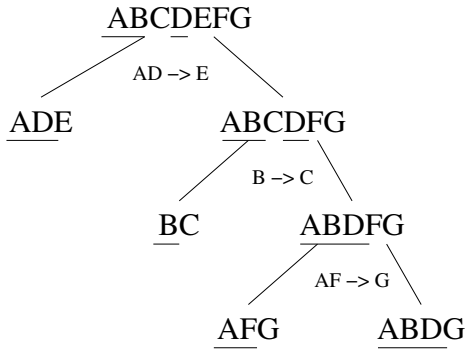
   a. (3 points) What are the candidate keys of **R**?

   There is just one: ABD.
   A, B, nor D appear on the right-hand side of a functional dependency. So each must be part of *any* candidate key. We can start then by checking out just ABD: $\{\mathsf{ABD}\}^+ = \{\mathsf{ABCDEFG}\}$. Thus ABD is a candidate key!
   Any other possible "key" must contain A, B, and D, and some other attribute. But this would be a super-key. So we are done.

$\mathsf{AD} \mapsto \mathsf{E}$ $\mathsf{BE} \mapsto \mathsf{F}$
$\mathsf{B} \mapsto \mathsf{C}$ $\mathsf{AF} \mapsto \mathsf{G}$

---

b. (4 points) Provide a BCNF lossless-join decomposition for **R** (ABCDEFG). Show your steps and your decomposition tree.



An answer then: ADE, BC, AFG, and ABDF.
We use those FDs that break BCNF with which to decompose. In this case, I chose $\mathsf{AD} \mapsto \mathsf{E}$ first. It breaks BCNF, since AD is not a key or superkey. The underlines indicate a candidate key for each relation along the way.
I use the basic lossless decomposition step to make one sub-relation of essentially the attributes of the FD I'm employing, and the other sub-relation of the attributes minus the right-hand side of the FD. So the intersection of the sub-relations is the left-hand side of the FD. And this is by design a candidate key of the first sub-relation. This means that the decomposition step is necessarily lossless.
We are not done. We check to see if our new relations, ADE and ABCDFG, are in BCNF. Technically, in each case, we have to determine the candidate keys again, and which of the FDs (from the closure set, $\mathcal{F}^+$, not just those we see!) apply. We can see that AD is the only candidate key of ADE, and only $\mathsf{AD} \mapsto \mathsf{E}$ applies. So it is in BCNF. We find ABD is the only candidate key of ABCDFG, and $\mathsf{B} \mapsto \mathsf{C}$ and $\mathsf{AF} \mapsto \mathsf{G}$ still apply. Both violate BCNF for ABCDFG. I chose to use $\mathsf{B} \mapsto \mathsf{C}$ for the next decomposition step, resulting in BC and ABDFG.
We check and find BC is in BCNF, but ABDFG is not. ABD is ABDFG's only candidate key, and only $\mathsf{AF} \mapsto \mathsf{G}$ applies. Using it to decompose, we get AFG and ABDG. Both are in BCNF, so we can stop.

$\mathsf{AD} \mapsto \mathsf{E}$ $\mathsf{BE} \mapsto \mathsf{F}$
$\mathsf{B} \mapsto \mathsf{C}$ $\mathsf{AF} \mapsto \mathsf{G}$

---

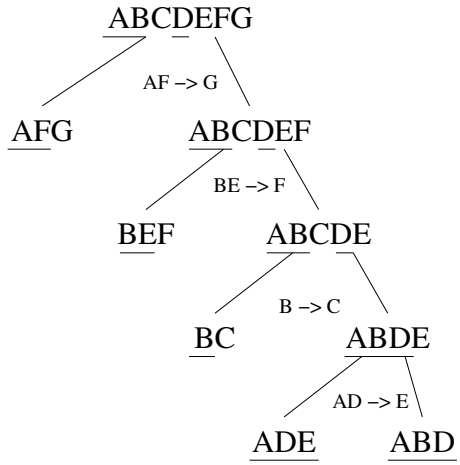
c. (3 points) Provide a BCNF decomposition that is lossless *and* is dependency preserving, or explain why there seems to be none in this case.

(If your answer to 5b is already dependency preserving, just say that here. Or possibly extend your answer to 5b: "My tables from 5b plus these extra ones.")

There might be such a decomposition. We only know that there does not *have to exist* any such. Let us look. Unfortunately, my decomposition in answer to Question 5b is not dependency preserving; $\mathsf{BE} \mapsto \mathsf{F}$ is not covered.

One solution is to try to add any non-covered FDs as relations to a BCNF, lossless decomposition that we have already devised. What can go wrong is these additions might not be in BCNF, causing us to further decompose, and we might lose coverage again! Let us see in this case: Add BEF. BEF is in BCNF since the only FD—from $\mathcal{F}^+$, remember—that applies is $\mathsf{BE} \mapsto \mathsf{F}$. We won. An answer then is ADE, BC, AFG, ABDF, and BEF.

Another approach is to find another BCNF, lossless decomposition tree that is also dependency preserving (and so does not require any "patches"). In a way, this is preferable, because there is less (controlled) redundancy in such a solution. However, there might not exist any. For this example, there does:



(Scratch space.)

More on 3a.
For the rel-ships advised_by, instance_of, belongs_to, and offered_by, we do not need to make tables, since these are many-one. We can handle each by a foreign key. For instance, to handle advised_by, we add an attribute advisor to table **Student** and a foreign key in **Student** referencing **Faculty** via that attribute onto **Faculty**'s primary key, fid.
In the case of advisor in **Student**, we allow it to be nullable, because a given student does not have to have an advisor. This mirrors that the many-one **advised_by** is marked as non-mandatory participation (by not being in bold).
When the many-one is marked as mandatory participation on the arrow side, we capture this by making the corresponding attributes in the "child" table (the one owning the foreign key) not nullable. For instance, a **Faculty** must belong to a **Dept**, so we mark the attribute dname in our table for **Faculty** as not nullable.
Any attributes owned by a many-one rel-ship become attributes in the child table; for example, eval is in table **Class**.
The many-many rel-ships **enrolled_in** and **can_teach** have to be made into tables.
Note that **Course** is a weak entity on **Dept**, and in turn, **Class** is a weak entity on **Course**.

STOP. BREATHE. RELAX. TURN IN YOUR EXAM.