DO NOT consult this solution set until after you have completed all work on your assignment. It is a violation of the Honor Code to view the solution set before your work is completed and turned in.

Assignment 6 – SOLUTIONS

It is an honor code violation to look at these solutions before completing and submitting the CS121 assignment. This is also true if you are not presently taking CS121, but may do so in the future.

If you have already completed and submitted your assignment, feel free to read on!

CS121 – Introduction to Relational Databases Fall 2014

DO NOT consult this solution set until after you have completed all work on your assignment. It is a violation of the Honor Code to view the solution set before your work is completed and turned in.

Problems

1. Here are four simple E-R diagrams. The entity-set schemas are $A(a)$ and $B(b)$, and R will obviously have the schema (a, b). For each diagram, specify what the non-trivial functional dependencies are, if any. If there are no functional dependencies, make sure you state that!

   (2 pts per part; 8 pts total)

   a) Many-to-many mapping between A and B

   

   Solution: No non-trivial functional dependencies.

   b) One-to-many mapping between A and B

   

   Solution: the functional dependency $b \rightarrow a$ holds.

   c) Many-to-one mapping between A and B

   

   Solution: the functional dependency $a \rightarrow b$ holds.

   d) One-to-one mapping between A and B

   

   Solution: two functional dependencies hold: $a \rightarrow b$, and $b \rightarrow a$.

2. For the Union rule, the Decomposition rule, and the Pseudotransitivity rule (pg. 339 in the textbook), use Armstrong's axioms to prove that each of these rules is sound. Make sure to specify what axiom(s) you apply for each step of your proofs. *(3 pts per rule; 9 pts total)*

   Solution:

   The **Union rule** states that if $\alpha \rightarrow \beta$ holds on R and $\alpha \rightarrow \gamma$ holds on R, then $\alpha \rightarrow \beta\gamma$ holds on R. Using only Armstrong's axioms, we can derive:

$\alpha \rightarrow \beta$	given
$\alpha \rightarrow \alpha\beta$	Augmentation
$\alpha \rightarrow \gamma$	given
$\alpha\beta \rightarrow \beta\gamma$	Augmentation
$\alpha \rightarrow \beta\gamma$	Transitivity

The **Decomposition rule** states that if $\alpha \rightarrow \beta\gamma$ holds on R then $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ also both hold on R. Using only Armstrong's axioms, we can derive:

$\alpha \rightarrow \beta\gamma$	given
$\beta\gamma \rightarrow \beta$	Reflexivity
$\alpha \rightarrow \beta$	Transitivity
$\beta\gamma \rightarrow \gamma$	Reflexivity
$\alpha \rightarrow \gamma$	Transitivity

The **Pseudotransitivity rule** states that if $\alpha \rightarrow \beta$ holds on R and $\gamma\beta \rightarrow \delta$ holds on R, then $\alpha\gamma \rightarrow \delta$ holds on R. Using only Armstrong's axioms, we can derive:

$\alpha \rightarrow \beta$	given
$\alpha\gamma \rightarrow \gamma\beta$	Augmentation
$\gamma\beta \rightarrow \delta$	given
$\alpha\gamma \rightarrow \delta$	Transitivity

3. (Practice Problem 8.6) Given the relation schema $R = (A, B, C, D, E)$, and this set of functional dependencies:

$$F = \{ A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A \}$$

a) Specify all candidate keys for R. You do not need to prove exhaustively that they are candidate keys, but make sure to show that each candidate key is a superkey for R.

Solution:

There are four candidate keys for R, given F: A, BC, CD, E

We can compute the attribute-set closure of each of these to show that it is a superkey:

A^+	$= A$	(start)
	$= ABC$	$(A \rightarrow BC)$
	$= ABCD$	$(B \rightarrow D)$
	$= ABCDE$	$(CD \rightarrow E)$

Since $E \rightarrow A$, it should be obvious that E^+ is also $ABCDE$.

Similarly, since $CD \rightarrow E$, CD^+ is also $ABCDE$. C^+ is simply C, and D^+ is D, so CD is also a candidate key.

Finally, BC^+ is as follows:

BC^+ $= BC$ (start)
$= BCD$ $(B \rightarrow D)$
$= BCDE$ $(CD \rightarrow E)$
$= ABCDE$ $(E \rightarrow A)$

B^+ is BD, and C^+ is C, as stated before, so BC is a candidate key.

b) Describe <u>all</u> functional dependencies that will appear in the closure F^+ of F. **This includes dependencies inferred from the above set, as well as trivial functional dependencies.** You can either describe F^+ by computing the entire set exhaustively (e.g. with a program you write yourself), or you can summarize the dependencies.

Solution:

There are 845 total dependencies in F^+, 211 of which are trivial functional dependencies. (Note that for a relation schema $R(A, B, C, D, E)$ with an empty set of functional dependencies $F = \{\}$, you automatically get 211 trivial functional dependencies, just due to Reflexivity.)

Because there are so many dependencies, it is unfruitful to attempt to list them unless you use a program to compute the entire closure of F.

Another approach would be to summarize the contents of F^+, given what we know about R and F, and also functional dependencies in general. F^+ will contain:

- All **trivial functional dependencies** $\alpha \rightarrow \beta$, where $\alpha \subseteq R$, and $\beta \subseteq \alpha$.
  (Generated by Reflexivity and Augmentation.)

- All **functional dependencies due to candidate keys**:
  $A\alpha \rightarrow \beta$, where $\alpha \subseteq \{BCDE\}$ and $\beta \subseteq R$.
  $BC\alpha \rightarrow \beta$, where $\alpha \subseteq \{ADE\}$ and $\beta \subseteq R$.
  $CD\alpha \rightarrow \beta$, where $\alpha \subseteq \{ABE\}$ and $\beta \subseteq R$.
  $E\alpha \rightarrow \beta$, where $\alpha \subseteq \{ABCD\}$ and $\beta \subseteq R$.

- All **functional dependencies generated by other non-superkey dependencies in *F***:
  $B \rightarrow D$, $B \rightarrow BD$

  These are only necessary to include because B is the only attribute that is on the LHS of a dependency, but is not a superkey of R. You may note that we don't include dependencies like $BD \rightarrow D$ etc., since these are covered by the first point above. Similarly, we don't include dependencies like $B\alpha \rightarrow$ …, since anything with a LHS including more than BD will be covered by the candidate-key FDs.

As another example, the book solution summarizes the answer this way:

Starting with $A \rightarrow BC$, we can conclude: $A \rightarrow B$ and $A \rightarrow C$.
Since $A \rightarrow B$ and $B \rightarrow D$, $A \rightarrow D$ (decomposition, transitive)
Since $A \rightarrow CD$ and $CD \rightarrow E$, $A \rightarrow E$ (union, decomposition, transitive)
Since $A \rightarrow A$, we have (reflexive)

$A \rightarrow ABCDE$ from the above steps (union)
Since $E \rightarrow A$, $E \rightarrow ABCDE$ (transitive)
Since $CD \rightarrow E$, $CD \rightarrow ABCDE$ (transitive)
Since $B \rightarrow D$ and $BC \rightarrow CD$, $BC \rightarrow ABCDE$ (augmentative, transitive)
Also, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow D$, etc.

Therefore, any functional dependency with A, E, BC, or CD on the left hand side of the arrow is in F^+, no matter which other attributes appear in the FD.

Allow $*$ to represent any set of attributes in R, then F^+ is $BD \rightarrow B$, $BD \rightarrow D$, $C \rightarrow C$, $D \rightarrow D$, $BD \rightarrow BD$, $B \rightarrow D$, $B \rightarrow B$, $B \rightarrow BD$, and all FDs of the form $A* \rightarrow \alpha$, $BC* \rightarrow \alpha$, $CD* \rightarrow \alpha$, $E* \rightarrow \alpha$ where α is any subset of $\{A, B, C, D, E\}$. The candidate keys are A, BC, CD, and E.

(10 points total)

4. Problem 8.33 *(5 points)*

Given a relational schema $R(A, B, C, D)$, does $A \rightarrow\!\!\!\!\rightarrow BC$ logically imply $A \rightarrow\!\!\!\!\rightarrow B$ and $A \rightarrow\!\!\!\!\rightarrow C$? If yes then prove it, otherwise give a counterexample.

(**Note:** $A \rightarrow\!\!\!\!\rightarrow BC$ is a multivalued dependency.)

Solution:

The multivalued dependency $A \rightarrow\!\!\!\!\rightarrow BC$ does not logically imply $A \rightarrow\!\!\!\!\rightarrow B$ or $A \rightarrow\!\!\!\!\rightarrow C$.

A trivial example illustrates that this is the case:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_1	c_1	d_2
a_1	b_2	c_2	d_1

This relation satisfies $A \rightarrow\!\!\!\!\rightarrow BC$, but it does not satisfy $A \rightarrow\!\!\!\!\rightarrow B$ or $A \rightarrow\!\!\!\!\rightarrow C$. (If $A \rightarrow\!\!\!\!\rightarrow BC$ in R, remember that $A \rightarrow\!\!\!\!\rightarrow D$ as well.) To satisfy $A \rightarrow\!\!\!\!\rightarrow B$ and $A \rightarrow\!\!\!\!\rightarrow C$, the relation would need four additional tuples:

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_2	d_2
a_1	b_1	c_1	d_2
a_1	b_2	c_2	d_1
a_1	b_1	c_2	d_1
a_1	b_2	c_1	d_2
a_1	b_1	c_2	d_2
a_1	b_2	c_1	d_1

Note that there are some interesting inference rules for multivalued dependencies that relate to this example. Namely, the **Multivalued union rule** states that if $\alpha \twoheadrightarrow \beta$ holds, and $\alpha \twoheadrightarrow \gamma$ holds, then $\alpha \twoheadrightarrow \beta\gamma$ also holds. From the above, we can clearly see that the inverse of this rule is not true.

5. Given a relation-schema $R(A, B, C, D, E, G)$, and the set of functional dependencies:

$F = \{ A \rightarrow E, BC \rightarrow D, C \rightarrow A, AB \rightarrow D, D \rightarrow G, BC \rightarrow E, D \rightarrow E, BC \rightarrow A \}$

a) Compute a canonical cover F_c of F. Show each step of your work:

- If you use inference rules on a step then note which rules on the step.
- If you assert that a particular attribute is extraneous in a dependency, show a proof that this is the case.

Failure to record the details of each step will result in point deductions.

Solution:

Start with:

$F_c = F = \{ A \rightarrow E, BC \rightarrow D, C \rightarrow A, AB \rightarrow D, D \rightarrow G, BC \rightarrow E, D \rightarrow E, BC \rightarrow A \}$

Step 1: Collapse down dependencies using Union rule

$F_c = \{ A \rightarrow E, C \rightarrow A, AB \rightarrow D, D \rightarrow EG, BC \rightarrow ADE \}$

Step 2: D is extraneous in $BC \rightarrow ADE$

Proof: Take *altered* set:

$F_c' = \{ A \rightarrow E, C \rightarrow A, AB \rightarrow D, D \rightarrow EG, BC \rightarrow AE \}$

See if we can infer $BC \rightarrow D$ from F_c'. Compute $\{BC\}^+ = ABCDE$. Result contains D, so D is extraneous in $BC \rightarrow ADE$.

Result: $F_c = \{ A \rightarrow E, C \rightarrow A, AB \rightarrow D, D \rightarrow EG, BC \rightarrow AE \}$

Step 3: B is extraneous in $BC \rightarrow AE$

Proof: Take *unaltered* set from step 2:

$F_c = \{ A \rightarrow E, C \rightarrow A, AB \rightarrow D, D \rightarrow EG, BC \rightarrow AE \}$

See if we can infer C → AE. Compute $\{C\}^+ = ACE$. Result contains AE, so B is extraneous in $BC \rightarrow AE$.

Result: $F_c = \{ A \rightarrow E, C \rightarrow A, AB \rightarrow D, D \rightarrow EG, C \rightarrow AE \}$

Step 4: $C \rightarrow A$ is extraneous due to $C \rightarrow AE$ (Decomposition Rule)

Result: $F_c = \{ A \rightarrow E, AB \rightarrow D, D \rightarrow EG, C \rightarrow AE \}$

Step 5: E is extraneous in $C \rightarrow AE$

Proof: Take *altered* set:

$F_c' = \{ A \rightarrow E, AB \rightarrow D, D \rightarrow EG, C \rightarrow A \}$

See if we can infer $C \rightarrow E$. Compute $\{C\}^+ = ACE$. Result contains E, so E is extraneous in $C \rightarrow AE$.

Result: $F_c = \{ A \rightarrow E, AB \rightarrow D, D \rightarrow EG, C \rightarrow A \}$

The final result is: $F_c = \{ A \rightarrow E, AB \rightarrow D, D \rightarrow EG, C \rightarrow A \}$

No two dependencies have the same left-hand side, and no dependency has any extraneous attributes.

b) Find a candidate key for R. Demonstrate that it is a superkey for R by computing its attribute-set closure. Then, demonstrate that it is a candidate key by computing the attribute-set closures of its proper subsets.

Solution:

BC is a candidate key.

$\{BC\}^+ = BC$	(initial state)
$\{BC\}^+ = ABC$	$C \rightarrow A$
$\{BC\}^+ = ABCDE$	$A \rightarrow E, AB \rightarrow D$
$\{BC\}^+ = ABCDEG$	$D \rightarrow EG$

BC only has two proper subsets, B and C.

$\{B\}^+ = B$	…and that's as far as we can go.
$\{C\}^+ = AC$	$C \rightarrow A$

BC is a minimal superkey; therefore it is a candidate key.

c) Decompose R into a BCNF schema. Prove that each of your final relation-schemas is indeed in BCNF. (You don't have to do this exhaustively, but make sure that you clearly demonstrate that each relation-schema is indeed in BCNF.)

Note all dependencies in F_c that are not preserved by this BCNF decomposition.

Solution:

1. Using $A \rightarrow E$, decompose R into $R_1(\underline{A}, E)$ and $R_{1B}(A, B, C, D, G)$.

   Is R_1 in BCNF? $A \rightarrow E$, and there are no nontrivial dependencies with E on the LHS. A is a key for R_1, so yes R_1 is in BCNF.

Is R_{1B} in BCNF? Obviously not, since there are several non-trivial dependencies that hold on R_{1B}. For example, $AB \rightarrow D$ and $C \rightarrow A$ both hold on R_{1B}. Need to decompose R_{1B}.

2. Using $AB \rightarrow D$, decompose $R_{1B}(A, B, C, D, G)$ into $R_2(\underline{A}, \underline{B}, D)$ and $R_{2B}(A, B, C, G)$.

   Is R_2 in BCNF? The only non-trivial dependency that holds on R_2 is $AB \rightarrow D$; all other non-trivial dependencies in F^+ fall outside of R_2. R_2 is in BCNF.

   Is R_{2B} in BCNF? Obviously not, since multiple nontrivial dependencies hold on R_{2B}. For example, $AB \rightarrow G$ (inferred from $AB \rightarrow D$, $D \rightarrow EG$) and $C \rightarrow A$ both hold on R_{2B}.

3. Using $AB \rightarrow G$, decompose R_{2B} into $R_3(\underline{A}, \underline{B}, G)$ and $R_4(A, B, C)$.

   Is R_3 in BCNF? The only non-trivial dependency that holds on R_3 is $AB \rightarrow G$, so R_3 is in BCNF.

   Is R_4 in BCNF? There are actually two non-trivial dependencies that hold on R_{3B}, $C \rightarrow A$ and $BC \rightarrow A$ (since BC is a candidate key of R). But, we know that B is extraneous in $BC \rightarrow A$ because of $C \rightarrow A$, so we know that R_4 is also in BCNF.

The final results are:

$R_1(\underline{A}, E)$
$R_2(\underline{A}, \underline{B}, D)$
$R_3(\underline{A}, \underline{B}, G)$
$R_4(\underline{C}, A, B)$

There is one dependency in F_c that isn't preserved: $D \rightarrow EG$

Note: It *appears* that we could combine steps 2 and 3 using $AB \rightarrow DG$, since this is implied by $AB \rightarrow D$ and $AB \rightarrow G$. If we did this, we would have a relation $(\underline{A}, \underline{B}, D, G)$, and this is not in BCNF because $D \rightarrow G$. So, further decompositions would be necessary, and you would get $(\underline{D}, G)$ and $(\underline{A}, \underline{B}, D)$. Just a different decomposition…

d) Find a second BCNF decomposition for R. (Hint: when you choose how to decompose a particular schema, you sometimes have multiple choices.) Again, prove that each of the final relation-schemas is indeed in BCNF. (If you already proved that a particular schema is in BCNF in part c, just write a note like, "R_i is in BCNF; see part c.") As before, note all dependencies in F_c that are not preserved by this BCNF decomposition.

Solution:

This time, start by decomposing R using our one un-preserved dependency from last time: $D \rightarrow EG$

1. Using $D \rightarrow EG$, decompose R into $R_1(\underline{D}, E, G)$ and $R_{1B}(A, B, C, D)$.

   Is R_1 in BCNF? The only non-trivial dependency that holds on R_1 is $D \rightarrow EG$. No non-trivial dependencies have E or G on the LHS.

Is R_{1B} in BCNF? Obviously not; for example, we have two nontrivial dependencies $AB \rightarrow D$ and $C \rightarrow A$ that hold on R_{1B}.

2. Using $C \rightarrow A$, decompose R_{1B} into $R_2(\underline{C}, A)$ and $R_3(\underline{B}, \underline{C}, D)$.

   Is R_2 in BCNF? The only non-trivial dependency that holds on R_2 is $C \rightarrow A$, so R_2 is in BCNF.

   Is R_3 in BCNF? The only non-trivial dependency that holds on R_3 is $BC \rightarrow D$ (inferred), so R_3 is also in BCNF.

The final results are:

$R_1(\underline{D}, E, G)$
$R_2(\underline{C}, A)$
$R_3(\underline{B}, \underline{C}, D)$

There are two dependencies in F_c that aren't preserved: $A \rightarrow E$, $AB \rightarrow D$

e) Using the 3NF schema synthesis algorithm, create a 3NF schema for R.

Solution:

We have already computed the cover of F: $F_c = \{ A \rightarrow E, AB \rightarrow D, D \rightarrow EG, C \rightarrow A \}$

Just run through the dependencies in left-to-right order:

$R_1(\underline{A}, E)$
$R_2(\underline{A}, \underline{B}, D)$
$R_3(\underline{D}, E, G)$
$R_4(\underline{C}, A)$

For each dependency $\alpha \rightarrow \beta$, $(\alpha \cup \beta)$ isn't contained within any previous schema R_i, so each dependency gets its own schema.

From part b, we computed that $BC \rightarrow R$, so BC is a candidate key for R. Because BC is not contained within any of these generated relations, we must add a fifth schema:

$R_5(\underline{B}, \underline{C})$

This is the complete 3NF decomposition of R.

(Part a: 8 pts; part b: 5 pts; part c: 8 pts; part d: 8 pts; part e: 6 pts. Total: 35 pts.)

6. As we saw in the previous problem, a particular schema can be normalized in several different ways, even when using only a single normal form. What actually makes the most sense depends on the enterprise that the schema is actually modeling.

   Consider this schema for a database of courses and sections: *R*(*course_id*, *section_id*, *dept*, *units*, *course_level*, *instructor_id*, *term*, *year*, *meet_time*, *room*, *num_students*). The department, term

and year are included, as is the instructor's ID, and the details of when and where each course is held. Of course, this schema is not normalized at all.

These functional dependencies also hold on *R*:

{ *course_id* } → { *dept*, *units*, *course_level* }
{ *course_id*, *section_id*, *term*, *year* } → { *meet_time*, *room*, *num_students*, *instructor_id* }
{ *room*, *meet_time*, *term*, *year* } → { *instructor_id*, *course_id*, *section_id* }

a) Find all candidate keys of *R*. Prove that each one is a superkey. (You don't have to prove that they are candidate keys; that would be annoying.)

Solution:

This database schema has two candidate keys.

{ *room*, *meet_time*, *term*, *year* }

The closure of this attribute-set is:

{ *room*, *meet_time*, *term*, *year* }	(initial)
{ *room*, *meet_time*, *term*, *year*, *instructor_id*, *course_id*, *section_id* }	(last dep.)
{ *room*, *meet_time*, *term*, *year*, *instructor_id*, *course_id*, *section_id*, *dept*, *units*, *course_level* }	(1st dep.)
{ *room*, *meet_time*, *term*, *year*, *instructor_id*, *course_id*, *section_id*, *dept*, *units*, *course_level*, *num_students* }	(2nd dep.)

{ *course_id*, *section_id*, *term*, *year* }

The closure of this attribute-set is:

{ *course_id*, *section_id*, *term*, *year* }	(initial)
{ *course_id*, *section_id*, *term*, *year*, *meet_time*, *room*, *num_students*, *instructor_id* }	(2nd dep.)
{ *course_id*, *section_id*, *term*, *year*, *meet_time*, *room*, *num_students*, *instructor_id*, *dept*, *units*, *course_level* }	(1st dep.)

b) Suggest a canonical cover F_c for the above set of functional dependencies. You might note that there are multiple options for F_c, so choose an F_c that is appropriate for what is being modeled by the schema and dependencies above. Explain your rationale.

Solution:

It is important to note that the set of functional dependencies above are not a canonical cover. The attribute *instructor_id* is extraneous in either the second dependency or the third dependency above; it can be removed from exactly one of these two dependencies. It makes the most sense to remove it from the third dependency, because instructors are usually associated with courses and sections, rather than with rooms and meeting times.

This gives us an F_c of:

{*course_id*} → {*dept*, *units*, *course_level*}
{*course_id*, *section_id*, *term*, *year*} → {*meet_time*, *room*, *num_students*, *instructor_id*}
{*room*, *meet_time*, *term*, *year*} → {*course_id*, *section_id*}

c) Suggest both a normal form and a schema decomposition that would be best for actual use in a course management system. Briefly explain your rationale. Make sure to indicate any and all primary, candidate, and foreign keys in your answer.

You should take into account both the scale of this database (i.e. how many records would be managed in such a database, and the practicality or impracticality of enforcing constraints in this database), as well as the need for ensuring a correct and complete representation in the database.

Solution:

Once the extraneous attribute is removed, a 3NF schema based on the cover would be quite acceptable. 3NF schemas have the potential to produce extra redundancy (and this one certainly would), but this is acceptable for two reasons:

1. Most universities have relatively small course catalogs. One can figure that there will probably be no more than 500-1000 total courses across all divisions and departments within the university[1], and the vast majority will have 20 or fewer sections. This gives us perhaps 20,000 records to keep track of; not a large database.

2. It is important to ensure correctness within this database system, so we would like to preserve all functional dependencies.

A 3NF schema would be:

courses(*course_id*, *dept*, *units*, *course_level*)

course_sections(*course_id*, *section_id*, *term*, *year*, *meet_time*, *room*, *instructor_id*, *num_students*)
- FK of *course_id* to *courses.course_id*
- FK of (*room*, *meet_time*, *term*, *year*) to *room_schedules*

room_schedules(*room*, *meet_time*, *term*, *year*, *course_id*, *section_id*)
- FK of (*course_id*, *section_id*, *term*, *year*) to *course_sections*

All of our functional dependencies are preserved by this decomposition. However, our *room_schedules* table is completely contained within *course_sections*, and we know that (*year*, *term*, *room*, *meet_time*) is also a candidate key. This suggests a better 3NF schema:

courses(*course_id*, *dept*, *units*, *course_level*)

section_details(*course_id*, *section_id*, *term*, *year*, *meet_time*, *room*, *instructor_id*, *num_students*)
- FK of *course_id* to *courses.course_id*
- Enforce (*year*, *term*, *room*, *meet_time*) as a candidate key

This schema eliminates even more redundancy from the first schema, and still preserves all functional dependencies.

(Part a: 6 pts; part b: 8 pts; part c: 6 pts. Total: 20 pts.)

[1] If this seems large to you, Caltech had 240 courses in the TQFR survey list for the Winter 2008 term alone. And this is a small school.

DO NOT consult this solution set until after you have completed all work on your assignment. It is a violation of the Honor Code to view the solution set before your work is completed and turned in.

7. Given the following database, and set of functional and multivalued dependencies:

   emails(*email_id*, *send_date*, *from_addr*, *to_addr*, *subject*, *email_body*, *attachment_name*, *attachment_body*)

   email_id → *send_date*, *from_addr*, *subject*, *email_body*
   email_id →→ *to_addr* (a multivalued dependency)
   email_id, *attachment_name* → *attachment_body*

   Create a set of 4NF relation schemas for the above database. Make sure to indicate all primary and foreign keys in your resulting schemas. Give each of your resulting schemas a descriptive name; don't just call them R_1, R_2, etc.

   Also, for each of your resulting schemas, demonstrate that each schema is in 4NF by citing the appropriate criteria from the 4NF normal form conditions, as well as indicating which of the above dependencies are relevant to your statements.

   (10 points)

   Solution:

   This problem is made easier by the fact that people generally understand how emails work and the constraints on email messages. Here are some good schemas to decompose the above into:

   email_info(*email_id*, *send_date*, *from_addr*, *subject*, *email_body*)
   - This schema is obviously in 4NF because it is also in BCNF: All functional dependencies that hold on this table have *email_id* on the LHS, and *email_id* is the primary key of this relation schema.

   email_recipients(*email_id*, *to_addr*)
   - This schema is in 4NF because the only multivalued dependency (*email_id* →→ *to_addr*) is trivial on this relation.
   - Of course, this is a set; in SQL, both attributes would be part of the primary key.
   - The *email_id* attribute has a foreign key reference to *email_info.email_id*.

   email_attachments(*email_id*, *attachment_name*, *attachment_body*)
   - The *email_id* attribute has a foreign key reference to *email_info.email_id*.
   - This schema is also obviously in 4NF because it is also in BCNF: the functional dependency that holds on this table haves (*email_id*, *attachment_name*) on its LHS, and this is the primary key of the table.