

1. We want to determine the presence of non-trivial functional dependencies given simple E-R diagrams.

- a. There are **no non-trivial functional dependencies** since it is a many-to-many relationship.
 - b. Many-to-one relationship between B & A, respectively $\Rightarrow b \rightarrow a$
 - c. Many-to-one relationship between A & B, respectively $\Rightarrow a \rightarrow b$
 - d. One-to-one relationship between A & B $\Rightarrow b \rightarrow a, a \rightarrow b$
-

2. Let $a = \alpha, B = \beta, y = \gamma, d = \delta$

- a. Union rule: If $a \rightarrow B$ holds & $a \rightarrow y$ holds, then $a \rightarrow By$ holds
 - i. Augmentation rule: If $a \rightarrow B$, then $aa \rightarrow aB$
 - ii. Augmentation rule: If $a \rightarrow y$, then $aB \rightarrow yB$
 - iii. Transitivity rule: If $aa \rightarrow aB$ & $aB \rightarrow yB$, then $aa \rightarrow yB$
 - iv. If $aa \rightarrow yB$, then **$a \rightarrow yB$** due to equivalence of aa & a
 - v. QED
 - b. Decomposition rule: If $a \rightarrow By$ holds, then $a \rightarrow B$ holds & $a \rightarrow y$ holds
 - i. Reflexivity rule: $By \rightarrow y, By \rightarrow B$ since $B, y \subseteq By$
 - ii. Transitivity rule: If $a \rightarrow By$ & $By \rightarrow y$, then **$a \rightarrow y$**
 - iii. Transitivity rule: If $a \rightarrow By$ & $By \rightarrow B$, then **$a \rightarrow B$**
 - iv. QED
 - c. Pseudotransitivity rule: If $a \rightarrow B$ holds & $yB \rightarrow d$ holds, then $ay \rightarrow d$ holds
 - i. Augmentation rule: If $a \rightarrow B$, then $ay \rightarrow By$
 - ii. Transitivity rule: If $ay \rightarrow By$ & $yB \rightarrow d$, then **$ay \rightarrow d$**
 - iii. QED
-

3. $R = (A, B, C, D, E)$ & $F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A\}$

- a. We want to specify all candidate keys for R & show that each candidate key is a superkey for R. We know a candidate key is a minimal superkey, so for some attribute-set X, if $X \rightarrow ABCDE$ then X is a superkey for R, implying that it is also a candidate key. Let us examine the following attribute-sets.
 - i. A:
 1. Decomposition: If $A \rightarrow BC$, then $A \rightarrow B$ & $A \rightarrow C$

2. Transitivity: If $A \rightarrow B$ & $B \rightarrow D$, then $A \rightarrow D$
 3. Union: If $A \rightarrow C$ & $A \rightarrow D$, then $A \rightarrow CD$
 4. Transitivity: If $A \rightarrow CD$ & $CD \rightarrow E$, then $A \rightarrow E$
 5. Reflexivity: $A \rightarrow A$ since $A \subseteq A$
 6. Union: With $A \rightarrow BC$ (plus 1-5), then $A \rightarrow ABCDE$
 7. A is a candidate key
 8. QED
- ii. E:
1. Transitivity: If $E \rightarrow A$ & $A \rightarrow ABCDE$, then $E \rightarrow ABCDE$
 2. E is a candidate key
 3. QED
- iii. CD:
1. Transitivity: If $CD \rightarrow E$ & $E \rightarrow ABCDE$, then $CD \rightarrow ABCDE$
 2. CD is a candidate key
 3. QED
- iv. BC:
1. Augmentation: If $B \rightarrow D$, then $BC \rightarrow CD$
 2. Transitivity: If $BC \rightarrow CD$ & $CD \rightarrow E$, then $BC \rightarrow ABCDE$
 3. BC is a candidate key
 4. QED
- v. Therefore, we have proved that the candidate keys for R are **A, E, CD, BC**.
- b. We want to describe all functional dependencies (trivial and non-trivial) that will appear in the closure F^+ of F. Let us summarize them below using the previously proved candidate keys along with other appropriate combinations.
- i. $A \rightarrow ABCDE$ and all dependencies generated from Decomposition
 - ii. $B \rightarrow D$ (given)
 - iii. $B \rightarrow B$ (By Reflexivity)
 - iv. $B \rightarrow BD$ (By Union)
 - v. $BD \rightarrow BD$ (By Augmentation or Reflexivity)
 - vi. $BD \rightarrow B$ (By Decomposition)
 - vii. $BD \rightarrow D$ (By Decomposition)
 - viii. $BC \rightarrow ABCDE$ and all dependencies generated from Decomposition
 - ix. $CD \rightarrow ABCDE$ and all dependencies generated from Decomposition
 - x. $C \rightarrow C$ (By Reflexivity)
 - xi. $D \rightarrow D$ (By Reflexivity)
 - xii. $E \rightarrow ABCDE$ and all dependencies generated from Decomposition

4. Given $R(A, B, C, D)$, we want to determine if $A \twoheadrightarrow BC$ logically implies $A \twoheadrightarrow B$ and $A \twoheadrightarrow C$ through proof or counterexample (where \twoheadrightarrow indicates multivalued dependency). It does not appear that any of the inference rules could be utilized to prove that the claim is true. So, using the formal and pictorial definition of multivalued dependencies, let us provide a counterexample through tables. Let us first establish a simple table that satisfies $A \twoheadrightarrow BC$:

Table 1	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>
<u>t1</u>	a	b	b	d
<u>t2</u>	a	c	c	e
<u>t3</u>	a	b	b	e
<u>t4</u>	a	c	c	d

By definition, we can see that Table 1 would indicate that $A \twoheadrightarrow BC$ holds:

- $t1[A] = t2[A] = t3[A] = t4[A]$
- $t1[BC] = t3[BC]$ AND $t2[BC] = t4[BC]$
- $t1[AD] = t4[AD]$ AND $t2[AD] = t3[AD]$ (where $R - BC = AD$)

Now let us try to modify Table 1 into Table 2 such that it satisfies $A \twoheadrightarrow BC$ and $A \twoheadrightarrow B$:

Table 2	<u>A</u>	<u>B</u>	<u>C</u>	<u>D</u>
<u>t1</u>	a	b	c	d
<u>t2</u>	a	c	c	e
<u>t3</u>	a	b	c	e
<u>t4</u>	a	c	c	d

Now, we can see that Table 2 still satisfies $A \twoheadrightarrow BC$:

- $t1[A] = t2[A] = t3[A] = t4[A]$
- $t1[BC] = t3[BC]$ AND $t2[BC] = t4[BC]$
- $t1[AD] = t4[AD]$ AND $t2[AD] = t3[AD]$ (where $R - BC = AD$)

Table 2 also satisfies $A \twoheadrightarrow B$:

- $t1[A] = t2[A] = t3[A] = t4[A]$
- $t1[B] = t3[B]$ AND $t2[B] = t4[B]$
- $t1[ACD] = t4[ACD]$ AND $t2[ACD] = t3[ACD]$ (where $R - B = ACD$)

However, Table 2 does not satisfy $A \twoheadrightarrow C$, thus providing a counterexample where $A \twoheadrightarrow BC$ and $A \twoheadrightarrow B$ hold true while $A \twoheadrightarrow C$ does not hold:

- $t1[A] = t2[A] = t3[A] = t4[A]$
- $t1[C] = t3[C]$ AND $t2[C] = t4[C]$
- **$t1[ABD] \neq t4[ABD]$ AND $t2[ABD] \neq t3[ABD]$ (where $R - BC = ABD$)**

Therefore, by counterexample, **the original claim is false.**

5. Given $R(A, B, C, D, E, G)$ and $F = \{A \twoheadrightarrow E, BC \twoheadrightarrow D, C \twoheadrightarrow A, AB \twoheadrightarrow D, D \twoheadrightarrow G, BC \twoheadrightarrow E, D \twoheadrightarrow E, BC \twoheadrightarrow A\}$
 - a. We want to compute a canonical cover F_c of F . We shall compute F_c after each step.
 - i. Union: If $D \twoheadrightarrow G$ & $D \twoheadrightarrow E$, then $D \twoheadrightarrow GE$
 1. Now we have, $F_c = \{A \twoheadrightarrow E, BC \twoheadrightarrow D, C \twoheadrightarrow A, AB \twoheadrightarrow D, D \twoheadrightarrow GE, BC \twoheadrightarrow E, BC \twoheadrightarrow A\}$
 - ii. Transitivity: If $BC \twoheadrightarrow A$ & $A \twoheadrightarrow E$, then $BC \twoheadrightarrow E$
 - iii. Therefore, $BC \twoheadrightarrow E$ is extraneous
 1. Now we have, $F_c = \{A \twoheadrightarrow E, BC \twoheadrightarrow D, C \twoheadrightarrow A, AB \twoheadrightarrow D, D \twoheadrightarrow GE, BC \twoheadrightarrow A\}$
 - iv. Augmentation: If $C \twoheadrightarrow A$, then $BC \twoheadrightarrow AB$
 - v. Transitivity: If $BC \twoheadrightarrow AB$ & $AB \twoheadrightarrow D$, then $BC \twoheadrightarrow D$
 - vi. Therefore, $BC \twoheadrightarrow D$ is extraneous
 1. Now we have, $F_c = \{A \twoheadrightarrow E, C \twoheadrightarrow A, AB \twoheadrightarrow D, D \twoheadrightarrow GE, BC \twoheadrightarrow A\}$
 - vii. $C \twoheadrightarrow A$ is logically implied by F (obvious)
 - viii. Therefore, $C \twoheadrightarrow A$ is extraneous in $BC \twoheadrightarrow A$
 1. Now we have, $F_c = \{A \twoheadrightarrow E, C \twoheadrightarrow A, AB \twoheadrightarrow D, D \twoheadrightarrow GE\}$
 - ix. Final computation: **$F_c = \{A \twoheadrightarrow E, C \twoheadrightarrow A, AB \twoheadrightarrow D, D \twoheadrightarrow GE\}$**
 - b. Now, we want to find a candidate key for R and demonstrate that it is a superkey for R by computing its attribute-set closure. We then want to demonstrate that it is a candidate key through computation of the attribute-set closures of its proper subsets. We will first test if BC is a superkey.
 - i. Let us compute the attribute-set closure for BC (BC^+):
 1. Start with $BC^+ = BC$
 2. $BC \twoheadrightarrow D$, so $BC^+ = BCD$

3. $BC \rightarrow D$ & $D \rightarrow G$, so $BC^+ = BCDG$ (Transitivity)
4. $BC \rightarrow E$, so $BC^+ = BCDEG$
5. $BC \rightarrow A$, so $BC^+ = ABCDEG$
6. Therefore, we have shown BC is a superkey
- ii. Now, let us compute the attribute-set closures of its subsets (B^+ , C^+)
 1. Start with $B^+ = B$
 - a. $B^+ = B$
 2. Start with $C^+ = C$
 - a. $C \rightarrow A$, so $C^+ = AC$
 - b. $C \rightarrow A$ & $A \rightarrow E$, so $C^+ = ACE$ (Transitivity)
 3. **BC is a candidate key** since B^+ and C^+ is not R .
- c. We want to decompose R into a BCNF schema and prove that each of the final relation-schemas is in BCNF. We know that all dependencies in F_c are not preserved by this BCNF decomposition. At this point, we have established that BC is a superkey for R . Following the process for BCNF decomposition, we arrive at the following by starting with $A \rightarrow E$:
 - i. $R_1 = (\underline{A}, E)$ - In BCNF since A is a primary key
 - ii. $R_2 = (R - E) = (A, B, C, D, G)$
 1. We know $C \rightarrow A$
 2. $R_2 = (\underline{C}, A)$ - In BCNF since C is a primary key
 - iii. $R_3 = (R - A) = (B, C, D, G)$
 1. We know $D \rightarrow G$
 2. $R_3 = (\underline{D}, G)$ - In BCNF since D is a primary key
 - iv. $R_4 = (R - G) = (\underline{B}, \underline{C}, D)$ - In BCNF since BC is primary key (already know that BC is a candidate key)
 - v. So, we arrive at the final decomposition:
 1. **$R_1 = (\underline{A}, E)$**
 2. **$R_2 = (\underline{C}, A)$**
 3. **$R_3 = (\underline{D}, G)$**
 4. **$R_4 = (\underline{B}, \underline{C}, D)$**
- d. We want to find a second BCNF decomposition for R (and prove that each of the final relation-schemas is in BCNF). We know that all dependencies in F_c are not preserved by this BCNF decomposition. Following the same process for BCNF decomposition, let us start with a different dependency $AB \rightarrow D$:
 - i. $R_1 = (\underline{A}, \underline{B}, D)$ - In BCNF since AB is a primary key
 - ii. $R_2 = (R - D) = (A, B, C, E, G)$
 1. We know $A \rightarrow E$
 2. $R_2 = (\underline{A}, E)$ - In BCNF since A is a primary key
 - iii. $R_3 = (R - E) = (A, B, C, G)$
 1. We know $C \rightarrow A$
 2. $R_3 = (\underline{C}, A)$ - In BCNF since C is a primary key

- iv. $R_4 = (R - A) = (\underline{B}, \underline{C}, G)$ - In BCNF since BC is a primary key (already know that BC is a candidate key)
 - 1. We know $BC \rightarrow D$ & $D \rightarrow G$, so by Transitivity, we know $BC \rightarrow G$
- v. So, we arrive at a slightly different decomposition:
 - 1. $R_1 = (\underline{A}, \underline{B}, D)$
 - 2. $R_2 = (\underline{A}, E)$
 - 3. $R_3 = (\underline{C}, A)$
 - 4. $R_4 = (\underline{B}, \underline{C}, G)$
- e. We want to create a 3NF schema for R using the 3NF schema synthesis algorithm. Following the 3NF synthesis algorithm, we start with our previously defined $F_c = \{A \rightarrow E, C \rightarrow A, AB \rightarrow D, D \rightarrow GE\}$. Note we have established that BC is a candidate key. We can arrive at the following:
 - i. We loop through each functional dependency in F_c :
 - 1. $R_1 = (\underline{A}, E)$ - In 3NF since A is a primary key
 - 2. $R_2 = (\underline{C}, A)$ - In 3NF since C is a primary key
 - 3. $R_3 = (\underline{A}, \underline{B}, D)$ - In 3NF since AB is a primary key
 - 4. $R_4 = (\underline{D}, \underline{G}, E)$ - In 3NF since DG is a primary key
 - ii. At this point, no schema contains the candidate key BC, so let us add another relation-schema with the candidate key:
 - 1. $R_5 = (\underline{B}, \underline{C})$ - In 3NF since BC is a candidate key
 - iii. So, we arrive at the following 3NF schema for R:
 - 1. $R_1 = (\underline{A}, E)$
 - 2. $R_2 = (\underline{C}, A)$
 - 3. $R_3 = (\underline{A}, \underline{B}, D)$
 - 4. $R_4 = (\underline{D}, \underline{G}, E)$
 - 5. $R_5 = (\underline{B}, \underline{C})$

- 6. Given schema $R(course_id, section_id, dept, units, course_level, instructor_id, term, year, meet_time, room, num_students)$. Given the following functional dependencies:
 - $\{course_id\} \rightarrow \{dept, units, course_level\}$
 - $\{course_id, section_id, term, year\} \rightarrow \{meet_time, room, num_students, instructor_id\}$
 - $\{room, meet_time, term, year\} \rightarrow \{instructor_id, course_id, section_id\}$
 - a. We want to find all of candidate keys of R and prove that each is a superkey (no need to prove that they are candidate keys ~ *thank you for this*)
 - i. Let $R(course_id, section_id, dept, units, course_level, instructor_id, term, year, meet_time, room, num_students) = R(C, S, D, U, L, I, T, Y, M, R, N)$
 - 1. $\{C\} \rightarrow \{DUL\}$
 - 2. $\{CSTY\} \rightarrow \{MRNI\}$
 - 3. $\{RMTY\} \rightarrow \{ICS\}$

- ii. Compute $(C)^+$:
 1. $(C)^+ = C$
 2. $C \rightarrow DUL$
 - a. $C \subseteq C$, so $(C)^+ = CDUL$
 3. No further updates since $CSTY$ and $RMTY$ are not subsets of $(C)^+$. So, C is not a candidate key.
 - iii. Compute $(CSTY)^+$:
 1. $(CSTY)^+ = CSTY$
 2. $CSTY \rightarrow MRNI$
 - a. $CSTY \subseteq CSTY$, so $(CSTY)^+ = MRNICSTY$
 3. $C \rightarrow DUL$
 - a. $C \subseteq MRNICSTY$, so $(CSTY)^+ = MRNICSTYDUL = CSDULITYMRN$
 4. $R \subseteq (CSTY)^+$, so $CSTY = (course_id, section_id, term, year)$ is a candidate key
 - iv. Compute $(RMTY)^+$:
 1. $(RMTY)^+ = RMTY$
 2. $RMTY \rightarrow ICS$
 - a. $RMTY \subseteq RMTY$, so $(RMTY)^+ = RMTYICS$
 3. $C \rightarrow DUL$
 - a. $C \subseteq RMTYICS$, so $(RMTY)^+ = RMTYICSDUL$
 4. $CSTY \rightarrow MRNI$
 - a. $CSTY \subseteq RMTYICSDUL$, so $(RMTY)^+ = RMTYICSDULN = CSDULITYMRN$
 5. $R \subseteq (RMTY)^+$, so $RMTY = (room, meet_time, term, year)$ is a candidate key
- b. We want to identify the two canonical covers for the above set of functional dependencies. We also want to identify the canonical cover that is the most appropriate.
- i. Let us compute F_c^1 :
 1. We can clearly determine that `instructor_id` is extraneous which allows us to arrive at the following possibility:
 - a. $\{course_id\} \rightarrow \{dept, units, course_level\}$
 - b. $\{course_id, section_id, term, year\} \rightarrow \{meet_time, room, num_students, instructor_id\}$
 - c. $\{room, meet_time, term, year\} \rightarrow \{course_id, section_id\}$
 - ii. Let us compute F_c^2 :
 1. Again, we can clearly determine that `instructor_id` is extraneous which allows us to arrive at the following possibility:
 - a. $\{course_id\} \rightarrow \{dept, units, course_level\}$
 - b. $\{course_id, section_id, term, year\} \rightarrow \{meet_time, room, num_students, instructor_id\}$

num_students}

c. $\{room, meet_time, term, year\} \rightarrow \{instructor_id, course_id, section_id\}$

- iii. In regards to which canonical cover is the most appropriate, it appears that F_c^1 is a better choice over F_c^2 since it seems more useful to know the instructor_id from information regarding a specific course and section itself (as opposed to getting the instructor_id from information regarding class/section time and location). If someone wanted to find a certain instructor_id, they would be more likely to “search” up the class rather than “searching” up the room and time.
- c. We want to suggest both a normal form and a schema decomposition that would be the best in this situation.
- i. Let us provide a normal form (3NF) starting with F_c^1 . Note that we simply turn each functional dependency into its own relation-schema:
1. R1 = (course_id, dept, units, course_level)
 - a. Primary key: course_id
 2. R2 = (course_id, section_id, term, year, meet_time, room, num_students, instructor_id)
 - a. Primary key: (course_id, section_id, term, year)
 - b. Foreign key: course_id
 - c. Candidate key: (room, meet_time, term, year)
 3. So, we arrive at the following:
 - a. R1 = (course_id, dept, units, course_level)
 - i. R1 = (C, D, U, L)
 - b. R2 = (course_id, section_id, term, year, meet_time, room, num_students, instructor_id)
 - i. R2 = (C, S, T, Y, M, R, N, I)
- ii. Let us provide a schema decomposition (BCNF) using F and R(course_id, section_id, dept, units, course_level, instructor_id, term, year, meet_time, room, num_students):
1. Set R1 = (course_id, dept, units, course_level)
 - a. Primary key: course_id
 - i. In BCNF
 2. R2 = (R - {dept, units, course_level}) = (course_id, section_id, instructor_id, term, year, meet_time, room, num_students, instructor_id)
 - a. Primary key: (course_id, section_id, term, year)
 - b. Foreign key: course_id
 - c. Candidate key: (room, meet_time, term, year)
 - i. In BCNF

3. So, we arrive at the following decomposition:
 - a. $R1 = (\text{course_id}, \text{dept}, \text{units}, \text{course_level})$
 - i. $R1 = (C, D, U, L)$
 - b. $R2 = (\text{course_id}, \text{section_id}, \text{term}, \text{year}, \text{meet_time}, \text{room}, \text{num_students}, \text{instructor_id})$
 - i. $R2 = (C, S, T, Y, M, R, N, I)$
- iii. We can clearly see that both processes produced the same schemas. We can determine that these would be best for actual use in a course management system because these schemas would efficiently, correctly, and completely represent the data considering the known dependencies. We should note that, relative to other databases, this database would seemingly have a low number of records which provides another support for these schemas. Also, enforcing the necessary constraints associated with this representation would be quite practical. For example, it would be reasonably practical to enforce that each unique `course_id` maps to a `section_id`, `term`, and `year`, while also enforcing that the unique combination (`course_id`, `section_id`, `term`, `year`) would map to the combination of (`meet_time`, `room`, `num_students`, `instructor_id`). Overall, these schema definitions are very logical and efficient and would be able to correctly enforce the appropriate dependencies.

7. Given the following database and dependencies:
 - a. `emails(email_id, send_date, from_addr, to_addr, subject, email_body, attachment_name, attachment_body)`
 - b. $\text{email_id} \rightarrow \text{send_date}, \text{from_addr}, \text{subject}, \text{email_body}$
 - c. $\text{email_id} \rightarrow\rightarrow \text{to_addr}$
 - d. $\text{email_id}, \text{attachment_name} \rightarrow \text{attachment_body}$

Let us perform 4NF decomposition on emails:

- Start with 7.b:
 - $R1 = (\text{email_id}, \text{send_date}, \text{from_addr}, \text{subject}, \text{email_body})$
 - Primary key: `email_id`
 - In 4NF since `email_id` is clearly a superkey (7.b dependency)
 - $R2 = (R - \{\text{send_date}, \text{from_addr}, \text{subject}, \text{email_body}\}) = (\text{email_id}, \text{to_addr}, \text{attachment_name}, \text{attachment_body})$
 - Choose $\text{email_id} \rightarrow\rightarrow \text{to_addr}$ (equivalent to $\text{email_id} \rightarrow \text{to_addr}$)
 - $R2 = (\text{email_id}, \text{to_addr})$
 - Primary key: (`email_id`, `to_addr`)

- Foreign key: email_id
 - In 4NF since email_id \twoheadrightarrow to_addr is a trivial multivalued dependency (7.c dependency)
 - R3 = (R - to_addr) = (email_id, attachment_name, attachment_body)
 - Primary key: (email_id, attachment_name)
 - Foreign key: email_id
 - In 4NF since email_id is clearly a superkey (7.d dependency)
- With the following redefinitions of our relation-schemas, we arrive at the following:
 - R1 = received_emails(email_id, send_date, from_addr, subject, email_body)
 - R2 = email_recipients(email_id, to_addr)
 - R3 = attachments(email_id, attachment_name, attachment_body)