CS2102
Database Systems

Last Lecture

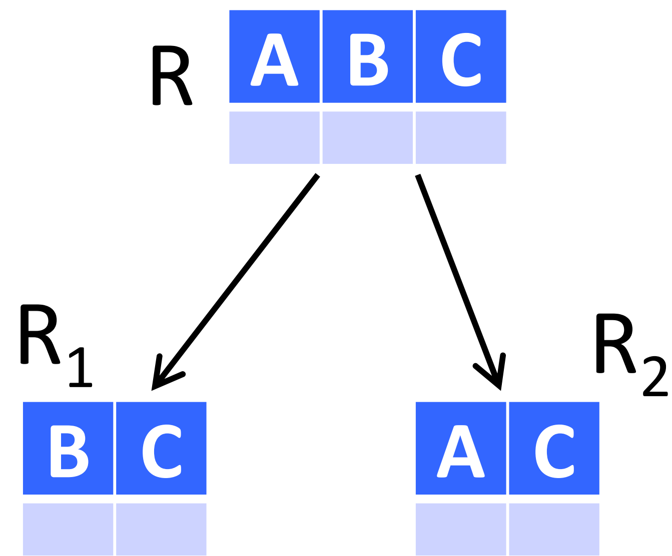
- BCNF Definition:
  - A table R is in BCNF, if every non-trivial and decomposed FD has a superkey as its left hand side
- BCNF Check:
  - Check if there exists a “more but not all” closure
  - E.g., a table R(X, Y, Z), with $\{X\}^+ = \{X, Y\}$
- BCNF Decomposition
  - If we have a table a table R(X, Y, Z), with $\{X\}^+ = \{X, Y\}$
  - Then decompose R into R1(X, Y) and R2(X, Z)
  - Repeat until all tables are in BCNF

Properties of BCNF Decomposition

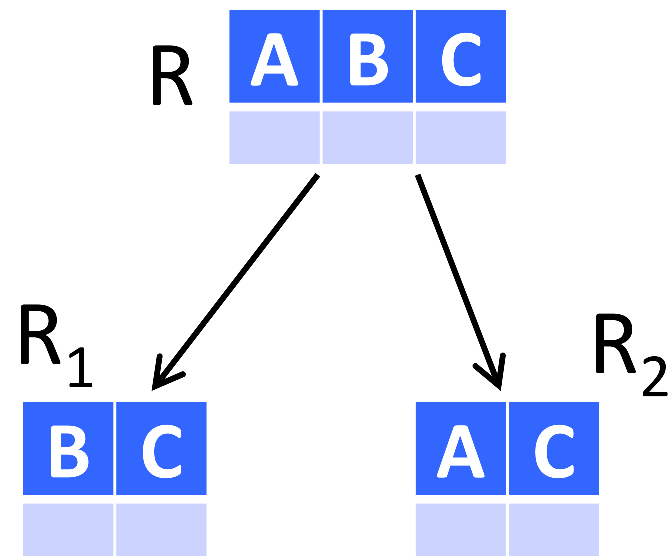
- Good properties
  - No update or deletion anomalies
  - Small redundancy
  - We can ensure that the original table can always be reconstructed from the decomposed tables
- Bad properties
  - Dependencies may not be preserved in the decomposed table

Dependency Preservation

- Given: Table R(A, B, C)
  - with AB→C, C→B
- Keys: {AB}, {AC}
- BCNF Decomposition
  - R_1(B, C)
  - R_2(A, C)
- Non-trivial and decomposed FDs on R_1: C→B
- Non-trivial and decomposed FDs on R_2: none
- The other FD, AB→C, cannot be derived from the FDs on R_1 and R_2, i.e., it is "lost"
- This why we say that a BCNF decomposition may not always preserve all FDs



Dependency Preservation

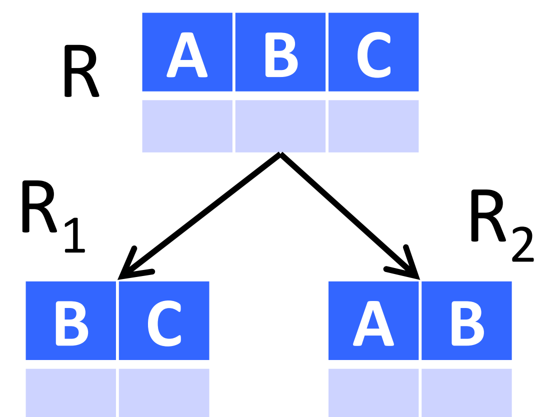


- What is the point of preserving FDs?
- It makes it easier to avoid "inappropriate" updates

- Previous example
  - We have two tables $R_1(B, C)$, $R_2(A, C)$
  - We have $C \rightarrow B$ and $AB \rightarrow C$
  - Due to $AB \rightarrow C$, we are not supposed to have two tuples (a1, b1, c1) and (a1, b1, c2)
  - But as we store A and C separately in R_1 and R_2, it is not easy to check whether such two tuples exist at the same time
  - That is, if someone wants to insert (a1, b1, c2), it is not easy for us to check whether (a1, b1, c1) already exists
  - This could be undesirable, depending on the application

Dependency Preservation

- Let S be the given set of FDs on the original table
- Let S' be the set of FDs on the decomposed tables
- We say that the decomposition preserves all FDs, if and only if S and S' are equivalent, i.e.,
  - Every FD in S' can be derived from S
  - Every FD in S can be derived from S'



- Example:
  - $S = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$
  - $S' = \{A \rightarrow B, B \rightarrow C\}$
  - S' can obviously be derived from S
  - S can also be derived from S', since
    $A \rightarrow B, B \rightarrow C \Longrightarrow A \rightarrow C$ (just check $\{A\}^+$ given S')
  - Hence, S and S' are equivalent

FD Equivalence: Example

- $S = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
- $S' = \{A \rightarrow CD, E \rightarrow AH\}$
- Prove that S and S' are equivalent
- First, prove that S' can be derived from S
  - Given S, we have $\{A\}^+ = \{ACD\}$, so $A \rightarrow CD$ is implied by S
  - Given S, we have $\{E\}^+ = \{EADHC\}$, so $E \rightarrow AH$ is implied by S
  - Hence, S' can be derived from S

FD Equivalence: Example

- $S = \{A \rightarrow C, AC \rightarrow D, E \rightarrow AD, E \rightarrow H\}$
- $S' = \{A \rightarrow CD, E \rightarrow AH\}$
- Prove that S and S' are equivalent
- Second, prove that S can be derived from S'
  - Given S', we have $\{A\}^+ = \{ACD\}$, so $A \rightarrow C$ is implied by S'
  - Given S', we have $\{AC\}^+ = \{ACD\}$, so $AC \rightarrow D$ is implied by S'
  - Given S', we have $\{E\}^+ = \{EADHC\}$, so $E \rightarrow AD$ and $E \rightarrow H$ are implied by S'
  - Hence, S can be derived from S'

Roadmap

- BCNF
  - Small redundancy
  - Lossless join property
  - But may not preserve all FDs
- Third Normal Form (3NF)
  - Not as strict as BCNF
  - Small redundancy (not as small as BCNF, though)
  - Lossless join property
  - Preserve all FDs

Third Normal Form (3NF)

- Definition: A table satisfies 3NF, if and only if for every non-trivial and decomposed FD
  - Either the left hand side is a superkey
  - Or the right hand side is a prime attribute (i.e., it appears in a key)

- Example:
  - Non-trivial and decomposed FDs: $C \rightarrow B$, $AC \rightarrow B$, $AB \rightarrow C$
  - Keys: {AB}, {AC}
  - $C \rightarrow B$ is OK, since B is a prime attribute
  - $AC \rightarrow B$ is OK, since AC is a key of R
  - $AB \rightarrow C$ is OK , since AB is a key of R
  - So R is in 3NF

R

A	B	C

Third Normal Form (3NF)

- Definition: A table satisfies 3NF, if and only if for every non-trivial and decomposed FD
  - Either the left hand side is a superkey
  - Or the right hand side is a prime attribute (i.e., it appears in a key)

- Another example:
  - Non-trivial and decomposed FDs: $A \rightarrow B$, $B \rightarrow C$, $AC \rightarrow B$, $AB \rightarrow C$
  - Keys: {A}
  - $A \rightarrow B$ is OK, since A is a superkey of R
  - $B \rightarrow C$ is not OK, since B is not a superkey of R, and C is not a prime attribute
  - So R is NOT in 3NF

R

A	B	C

BCNF vs. 3NF

- BCNF: For any non-trivial and decomposed FD,
  - The left hand side is a super-key

"Every attribute must depend ONLY on superkeys!"

"No exception!"

- 3NF: For any non-trivial and decomposed FD,
  - Either the left hand side is a super-key
  - Or the right hand side is a prime attribute

"Exceptions can be made for prime attributes."

BCNF vs. 3NF

- BCNF: For any non-trivial and decomposed FD,
  - The left hand side is a super-key
- 3NF: For any non-trivial and decomposed FD,
  - Either the left hand side is a super-key
  - Or the right hand side is a prime attribute

- 3NF is more "lenient" than BCNF
- Therefore,
  - Satisfying BCNF ==> satisfying 3NF, but not necessarily vice versa
  - Violating 3NF ==> violating BCNF, but not necessarily vice versa

3NF Check

- Input: a table R

1. Derive the keys of R
2. For each given FD, check if
   - The left hand side is a superkey, or
   - Each attribute on the right hand side is a prime attribute
3. If all given FDs satisfy this condition, then R is in 3NF

3NF Check: Example

- R(A, B, C, D) with FDs AB → C, C → D, and D→A
  1. Compute the closure for each attribute subset

  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{B\}$, $\{C\}^+ = \{ACD\}$, $\{D\}^+ = \{AD\}$
  - $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{ACD\}$, $\{AD\}^+ = \{AD\}$
  - $\{BC\}^+ = \{ABCD\}$, $\{BD\}^+ = \{ABCD\}$, $\{CD\}^+ = \{ACD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{BCD\}^+ = \{ABCD\}$
  - $\{ACD\}^+ = \{ACD\}$
  - $\{ABCD\}^+ = \{ABCD\}$

3NF Check: Example

- R(A, B, C, D) with FDs AB → C, C → D, and D→A

  2. Derive the keys of R

  Keys: AB, BC, BD

  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{B\}$, $\{C\}^+ = \{ACD\}$, $\{D\}^+ = \{AD\}$
  - $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{ACD\}$, $\{AD\}^+ = \{AD\}$
  - $\{BC\}^+ = \{ABCD\}$, $\{BD\}^+ = \{ABCD\}$, $\{CD\}^+ = \{ACD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{BCD\}^+ = \{ABCD\}$
  - $\{ACD\}^+ = \{ACD\}$
  - $\{ABCD\}^+ = \{ABCD\}$

3NF Check: Example

- R(A, B, C, D) with FDs AB → C, C → D, and D→A

  2. Derive the keys of R

  Keys: AB, BC, BD

  3. For each given FD, check if

     - The left hand side is a superkey, or
     - Each attribute on the right hand side is a prime attribute

In 3NF

Exercise: 3NF Check

- R(A, B, C, D) with FDs B → C, B → D

Exercise: 3NF Check

- R(A, B, C, D) with FDs B $\rightarrow$ C, B $\rightarrow$ D

  1. Compute the closure for each subset of the attributes in R

  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{BCD\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
  - $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{AC\}$, $\{AD\}^+ = \{AD\}$
  - $\{BC\}^+ = \{BCD\}$, $\{BD\}^+ = \{BCD\}$, $\{CD\}^+ = \{CD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{ABCD\}$
  - $\{BCD\}^+ = \{BCD\}$, $\{ACD\}^+ = \{ACD\}$
  - $\{ABCD\}^+ = \{ABCD\}$

Exercise: 3NF Check

- R(A, B, C, D) with FDs B $\rightarrow$ C, B $\rightarrow$ D

  2. Derive the keys of R

  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{BCD\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
  - $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{AC\}$, $\{AD\}^+ = \{AD\}$
  - $\{BC\}^+ = \{BCD\}$, $\{BD\}^+ = \{BCD\}$, $\{CD\}^+ = \{CD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{ABCD\}$
  - $\{BCD\}^+ = \{BCD\}$, $\{ACD\}^+ = \{ACD\}$
  - $\{ABCD\}^+ = \{ABCD\}$

Exercise: 3NF Check

- R(A, B, C, D) with FDs $B \rightarrow C$, $B \rightarrow D$

  2. Derive the keys of R

     keys: AB

  - $\{A\}^+ = \{A\}$, $\{B\}^+ = \{BCD\}$, $\{C\}^+ = \{C\}$, $\{D\}^+ = \{D\}$
  - $\{AB\}^+ = \{ABCD\}$, $\{AC\}^+ = \{AC\}$, $\{AD\}^+ = \{AD\}$
  - $\{BC\}^+ = \{BCD\}$, $\{BD\}^+ = \{BCD\}$, $\{CD\}^+ = \{CD\}$
  - $\{ABC\}^+ = \{ABD\}^+ = \{ABCD\}$
  - $\{BCD\}^+ = \{BCD\}$, $\{ACD\}^+ = \{ACD\}$
  - $\{ABCD\}^+ = \{ABCD\}$

Exercise: 3NF Check

- R(A, B, C, D) with FDs B $\rightarrow$ C, B $\rightarrow$ D

  2. Derive the keys of R

     keys: AB

  3. For each given FD, check if

     - The left hand side is a superkey, or
     - Each attribute on the right hand side is a prime attribute

Not in 3NF

Exercise: 3NF Check

- R(A, B, C, D) with FDs A $\rightarrow$ B, B $\rightarrow$ C, C$\rightarrow$D, and D$\rightarrow$A

Exercise: 3NF Check

In 3NF

- R(A, B, C, D) with FDs A → B, B → C, C→D, and D→A

  1. Compute the closure for each subset of the attributes in R
     - $\{A\}^+ = \{ABCD\}$, $\{B\}^+ = \{ABCD\}$, $\{C\}^+ = \{ABCD\}$, $\{D\}^+ = \{ABCD\}$
     - The others are all {ABCD}
  2. Find the keys: A, B, C, D
  3. For each given FD, check if
     - The left hand side is a superkey, or
     - Each attribute on the right hand side is a prime attribute

Exercise: 3NF Check

- R(A, B, C, D, E) with FDs AB → C, DE → C, B→E

Exercise: 3NF Check

- R(A, B, C, D, E) with FDs AB $\rightarrow$ C, DE $\rightarrow$ C, B$\rightarrow$E

  1. Compute the closure for each subset of the attributes in R
  2. Derive the keys of R
  3. For each given FD, check if
     - The left hand side is a superkey, or
     - Each attribute on the right hand side is a prime attribute
  4. If all given FDs satisfy this condition, then R is in 3NF

Exercise: 3NF Check

- R(A, B, C, D, E) with FDs AB → C, DE → C, B→E

1. Compute the closure for each subset of the attributes in R
2. Derive the keys of R
   - Notice that A, B, and D do not appear in the r.h.s. of any FD
   - So all keys must contain ABD
   - $\{ABD\}^+ = \{ABCDE\}$, so ABD is the only key
3. For each given FD, check if
   - The left hand side is a superkey, or
   - Each attribute on the right hand side is a prime attribute
4. If all given FDs satisfy this condition, then R is in 3NF

Exercise: 3NF Check

- R(A, B, C, D, E) with FDs AB → C, DE → C, B→E

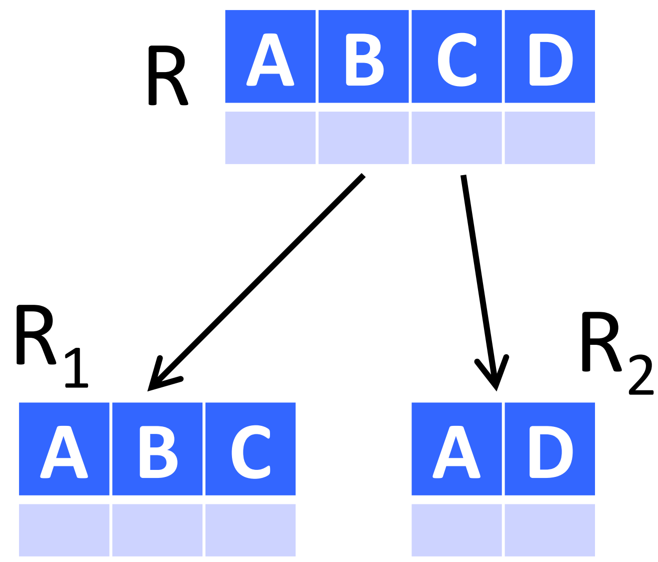
1. Compute the closure for each subset of the attributes in R
2. Derive the keys of R
   - ABD is the only key

Not in 3NF

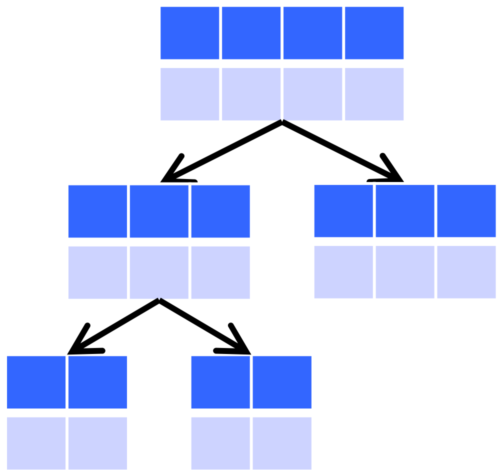
3. For each given FD, check if
   - The left hand side is a superkey, or
   - Each attribute on the right hand side is a prime attribute
4. If all given FDs satisfy this condition, then R is in 3NF

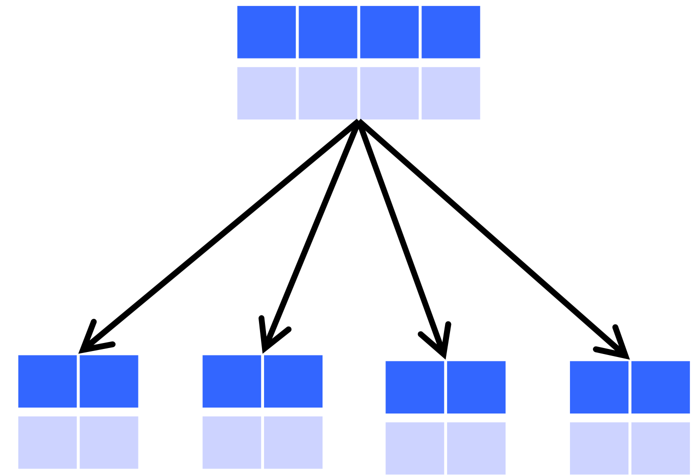
3NF Decomposition

- Given: A table NOT in 3NF
- Objective: Decompose it into smaller tables that are in 3NF
- Example
  - Given: R(A, B, C, D)
  - FDs: AB→C, C→B, A→D
  - Keys: {AB}, {AC}
  - R is not in 3NF, due to A→D
  - 3NF decomposition of R: R_1(A, B, C), R_2(A, D)



BCNF Decomposition vs. 3NF Decomposition



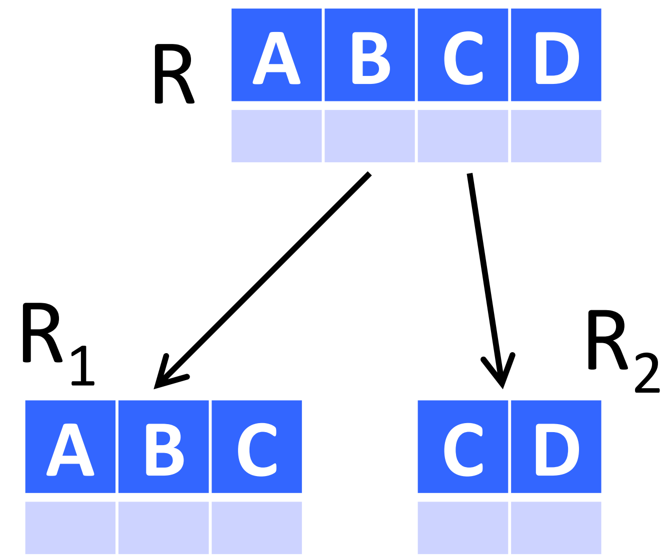


- A BCNF decomposition may perform one or more binary splits, each of which divides a table into two

- A 3NF decomposition has only one split, which divides the table into two or more parts

3NF Decomposition Algorithm

- Given: A table R, and a set S of FDs
  - e.g., R(A, B, C, D)
    $S = \{A \rightarrow BD, AB \rightarrow C, C \rightarrow D, BC \rightarrow D\}$
- Step 1: Derive a minimal basis of S
  - e.g., a minimal basis of S is
    $\{A \rightarrow B, A \rightarrow C, C \rightarrow D\}$
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
  - e.g., after combining $A \rightarrow B$ and $A \rightarrow C$,
    we have $\{A \rightarrow BC, C \rightarrow D\}$
- Step 3: Create a table for each FD remained
  - $R_1(A, B, C)$, $R_2(C, D)$
- Step 4: If none of the tables contains a key of the original table R, create a table that contains a key of R (any key would do)



- Step 5: Remove subsumed tables

Minimal Basis

- Given a set S of FDs, the minimal basis of S is a simplified version of S
  - Also called the minimal cover of S
- Previous example:
  - S = {A→BD, AB→C, C→D, BC→D}
  - A minimal basis: {A→B, A→C, C→D}
- How simplified?
- Four conditions.
- Condition 1: Every FD in the minimal basis can be derived from S, and vice versa.

Minimal Basis

- Previous example:
  - S = {A→BD, AB→C, C→D, BC→D}
  - A minimal basis: {A→B, A→C, C→D}
- Condition 2: Every FD in the minimal basis is a non-trivial and decomposed FD.
- Example in S: A→BD does not satisfy this condition
- That is why A→BD is not in the minimal basis

Minimal Basis

- Previous example:
  - S = {A→BD, AB→C, C→D, BC→D}
  - A minimal basis: {A→B, A→C, C→D}
- Condition 3: No FD in the minimal basis is redundant.
- That is, no FD in the minimal basis can be derived from the other FDs in the minimal basis.
- Example in S: BC→D can be derived from C→D
- That is why BC→D is not in the minimal basis

Minimal Basis

- Previous example:
  - S = {A→BD, AB→C, C→D, BC→D}
  - A minimal basis: {A→B, A→C, C→D}
- Condition 4: For each FD in the minimal basis, none of the attributes on the left hand side is redundant
- That is, if we remove an attribute from the left hand side, then the resulting FD is a new FD that cannot be derived from the original set of FDs
- Example:
  - Consider AB→C
  - If we remove B from the left hand side, we have A→C
  - A→C can be derived from S, since $\{A\}^+ = \{ABDC\}$ given S
  - This indicates that A→C is "hidden" in S
  - There, we can add A→C into S, without introducing extraneous information
  - Once A→C is added, AB→C becomes redundant and can be removed
  - Effectively, this indicates that B is redundant in AB→C
  - This is why AB→C is not in the minimal basis

Minimal Basis: Conditions

- Let S be a set of FDs
- Its minimal basis M is a set of FDs, such that
  1. every FD in S can be derived from M, and vice versa
  2. every FD in M is a non-trivial and decomposed FD
  3. if any FD is removed from M, then some FD in S cannot be derived from M
  4. for any FD in M, if we remove an attribute from its left hand side, then the FD cannot be derived from S

Minimal Basis: Example

- $S = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$, $M = \{A \rightarrow B, B \rightarrow C\}$
- M is a minimal basis of S
  1. every FD in S can be derived from M, and vice versa
  2. every FD in M is a non-trivial and decomposed FD
  3. if any FD is removed from M, then some FD in S cannot be derived from M
  4. for any FD in M, if we remove an attribute from its left hand side, then the FD cannot be derived from S

Minimal Basis: Example 2

- S = {A→B, B→C, A→C}, M = {A→B, AB→C}
- Is M a minimal basis of S?

1. every FD in S can be derived from M, and vice versa
2. every FD in M is a non-trivial and decomposed FD
3. if any FD is removed from M, then some FD in S cannot be derived from M
4. for any FD in M, if we remove an attribute from its left hand side, then the FD cannot be derived from S

This condition is not satisfied

Minimal Basis: Example 3

- S = {A→B, B→C, A→C}, M = {A→BC, B→C}
- Is M a minimal basis of S?
  1. every FD in S can be derived from M, and vice versa
  2. every FD in M is a non-trivial and decomposed FD
  3. if any FD is removed from M, then some FD in S cannot be derived from M
  4. for any FD in M, if we remove an attribute from its left hand side, then the FD cannot be derived from S

This condition is not satisfied

Minimal Basis: Example 4

- S = {A→B, B→C, A→C}, M = {A→B, B→C, A→C}
- Is M a minimal basis of S? This condition is not satisfied
  1. every FD in S can be derived from M, and vice versa
  2. every FD in M is a non-trivial and decomposed FD
  3. if any FD is removed from M, then some FD in S cannot be derived from M
  4. for any FD in M, if we remove an attribute from its left hand side, then the FD cannot be derived from S

Minimal Basis: Example 5

- S = {A→B, A→C, C→B}, M = {A→B, AB→C, C→B}
- Is M a minimal basis of S?
  1. every FD in S can be derived from M, and vice versa
  2. every FD in M is a non-trivial and decomposed FD
  3. if any FD is removed from M, then some FD in S cannot be derived from M
  4. for any FD in M, if we remove an attribute from its left hand side, then the FD cannot be derived from S

This condition is not satisfied

3NF Decomposition Algorithm

- Given: A table R, and a set S of FDs
  - Step 1: Derive a minimal basis of S
  - Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
  - Step 3: Create a table for each FD remained
  - Step 4: If none of the tables contain a key of the original table R, create a table that contains a key of R
  - Step 5: Remove subsumed tables
- How to find the minimal basis of S?
- Solution: start from the FDs on R, and then simplify it step by step

Algorithm for Minimal Basis

- Step 1: Transform the FDs, so that each right hand side contains only one attribute
- Step 2: Remove redundant attributes on the left hand side of each FD
- Step 3: Remove redundant FDs

Algorithm for Minimal Basis: Example

- Given: a set S of FDs
- Example: S = {A→BD, AB→C, C→D, BC→D}
- Step 1: Transform the FDs, so that each right hand side contains only one attribute
- Result: S = {A→B, A→D, AB→C, C→D, BC→D}
- Reason:
  - Condition 2 for minimal basis:
    Each FD is a non-trivial and decomposed FD

Algorithm for Minimal Basis: Example

- Result of the previous step:
  - S = {A→B, A→D, AB→C, C→D, BC→D}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Both AB→C and BC→D have more than one attribute on the lhs
- Let's check AB→C first
- Is A redundant?
- If we remove A, then AB→C becomes B→C
- Whether this removal is OK depends on whether B→C is implied by S
  - If B→C is implied by S, then the removal of A is OK, since the removal does not add extraneous information into S
- Is B→C implied by S?
- Check: Given S, we have $\{B\}^+$ = {B}, which does NOT contain C
- Therefore, B→C is not implied by S, and hence, A is NOT redundant

Algorithm for Minimal Basis: Example

- Result of the previous step:
  - S = {A→B, A→D, AB→C, C→D, BC→D}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Both AB→C and BC→D have more than one attribute on the lhs
- Let's check AB→C first
- Is B redundant?
- If we remove B, then AB→C becomes A→C
- Whether this is OK depends on whether A→C is implied by S
- Is A→C implied by S?
- Check: Given S, we have $\{A\}^+$ = {ABCD}, which contains C
- Therefore, A→C is implied by S, and hence, B is redundant in AB→C
- Thus, we can simplify AB→C to A→C
- Result: S = {A→B, A→D, A→C, C→D, BC→D}

Algorithm for Minimal Basis: Example

- Result of the previous step:
  - S = {A→B, A→D, A→C, C→D, BC→D}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Now let's check BC→D
- Is B redundant?
- If we remove B, then BC→D becomes C→D
- Whether this is OK depends on whether C→D is implied by S
- Is C→D implied by S?
- Yes, it is explicitly in S already
- Therefore, C→D is implied by S, and hence, B is redundant
- Thus, we can simplify BC→C to C→D
- Result: S = {A→B, A→D, A→C, C→D}
- Now there is no redundant attribute on the left hand side of any FD

Algorithm for Minimal Basis: Example

- Result of the previous step:
- S = {A→B, A→D, A→C, C→D}
- Step 3: Remove redundant FDs
- Is A→B redundant?
- i.e., is A→B implied by other FDs in S?
- Let's check
- Without A→B, we have {A→D, A→C, C→D}
- Given those FDs, we have $\{A\}^+$ = {ACD}, which does not contain B
- Therefore, A→B is not implied by the other FDs

Algorithm for Minimal Basis: Example

- Result of the previous step:
- S = {A→B, A→D, A→C, C→D}
- Step 3: Remove redundant FDs
- Is A→D redundant?
- i.e., is A→D implied by other FDs in S?
- Let's check
- Without A→D, we have {A→B, A→C, C→D}
- Given those FDs, we have $\{A\}^+ = \{ABCD\}$, which contains D
- Therefore, A→D is implied by the other FDs
- Hence, A→D is redundant and should be removed
- Result: S = {A→B, A→C, C→D}

Algorithm for Minimal Basis: Example

- Result of the previous step:
- S = {A→B, A→C, C→D}
- Step 3: Remove redundant FDs
- Is A→C redundant?
- i.e., is A→C implied by other FDs in S?
- Let's check
- Without A→C, we have {A→B, C→D}
- Given those FDs, we have $\{A\}^+ = \{AB\}$, which does not contain C
- Therefore, A→C is not implied by the other FDs

Algorithm for Minimal Basis: Example

- Result of the previous step:
- S = {A→B, A→C, C→D}
- Step 3: Remove redundant FDs
- Is C→D redundant?
- i.e., is C→D implied by other FDs in S?
- Let's check
- Without C→D, we have {A→B, A→C}
- Given those FDs, we have $\{C\}^+ = \{C\}$, which does not contain D
- Therefore, C→D is not implied by the other FDs
- Final minimal basis: {A→B, A→C, C→D}

Algorithm for Minimal Basis: Example 2

- Given: S = {BC$\rightarrow$DE, A$\rightarrow$E, D$\rightarrow$A, E$\rightarrow$B}

Algorithm for Minimal Basis: Example 2

- Given: S = {BC→DE, A→E, D→A, E→B}
- Step 1: Transform the FDs, so that each right hand side contains only one attribute
- Result: S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Both BC→D and BC→E have more than one attributes on the left hand side

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Let's check BC→D first
- Is B redundant?
- If we remove B, then BC→D becomes C→D
- Whether this removal is OK depends on whether C→D is implied by S
- Is C→D implied by S?
- Check: Given S, we have $\{C\}^+ = \{C\}$, which does NOT contain D
- Therefore, C→D is not implied by S, and hence, B is NOT redundant

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Let's check BC→D first
- Is C redundant?
- If we remove C, then BC→D becomes B→D
- Whether this removal is OK depends on whether B→D is implied by S
- Is B→D implied by S?
- Check: Given S, we have $\{B\}^+ = \{B\}$, which does NOT contain D
- Therefore, B→D is not implied by S, and hence, C is NOT redundant

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Now let's check BC→E
- Is B redundant?
- If we remove B, then BC→E becomes C→E
- Whether this removal is OK depends on whether C→E is implied by S
- Is C→E implied by S?
- Check: Given S, we have $\{C\}^+ = \{C\}$, which does NOT contain E
- Therefore, C→E is not implied by S, and hence, B is NOT redundant

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 2: Remove redundant attributes on the left hand side of each FD
- Now let's check BC→E
- Is C redundant?
- If we remove C, then BC→E becomes B→E
- Whether this removal is OK depends on whether B→E is implied by S
- Is B→E implied by S?
- Check: Given S, we have $\{B\}^+ = \{B\}$, which does NOT contain E
- Therefore, B→E is not implied by S, and hence, C is NOT redundant
- So there is no redundant attribute on the left hand side of any FD

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 3: Remove redundant FDs
- Is BC→D redundant?
- i.e., is BC→D implied by other FDs in S?
- Let's check
- Without BC→D, we have {BC→E, A→E, D→A, E→B}
- Given those FDs, we have $\{BC\}^+ = \{BCE\}$, which does not contain D
- Therefore, BC→D is not implied by the other FDs

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, BC→E, A→E, D→A, E→B}
- Step 3: Remove redundant FDs
- Is BC→E redundant?
- i.e., is BC→E implied by other FDs in S?
- Let's check
- Without BC→E, we have {BC→D, A→E, D→A, E→B}
- Given those FDs, we have $\{BC\}^+$ = {ABCDE}, which contains E
- Therefore, BC→E is implied by the other FDs, and can be removed
- Result: S = {BC→D, A→E, D→A, E→B}

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, A→E, D→A, E→B}
- Step 3: Remove redundant FDs
- Is A→E redundant?
- i.e., is A→E implied by other FDs in S?
- Let's check
- Without A→E, we have {BC→D, D→A, E→B}
- Given those FDs, we have $\{A\}^+ = \{A\}$, which does not contain E
- Therefore, A→E is not implied by the other FDs

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, A→E, D→A, E→B}
- Step 3: Remove redundant FDs
- Is D→A redundant?
- i.e., is D→A implied by other FDs in S?
- Let's check
- Without D→A, we have {BC→D, A→E, E→B}
- Given those FDs, we have $\{D\}^+ = \{D\}$, which does not contain A
- Therefore, D→A is not implied by the other FDs

Algorithm for Minimal Basis: Example 2

- Result of the previous step:
- S = {BC→D, A→E, D→A, E→B}
- Step 3: Remove redundant FDs
- Is E→B redundant?
- i.e., is E→B implied by other FDs in S?
- Let's check
- Without E→B, we have {BC→D, A→E, D→A}
- Given those FDs, we have $\{E\}^+ = \{E\}$, which does not contain B
- Therefore, E→B is not implied by the other FDs
- So the final minimal basis is: {BC→D, A→E, D→A, E→B}

Exercise

- $S = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

1. Transform the FDs to ensure that the right hand side of each FD has only one attribute
2. Check if we can remove any attribute from the left hand side of any FD
3. See if any FD can be derived from the other FDs. Remove those FDs one by one

Exercise

- S = {A→C, AC→D, AD→B}

1. Transform the FDs to ensure that the right hand side of each FD has only one attribute

- Result: S = {A→C, AC→D, AD→B}

2. Check if we can remove any attribute from the left hand side of any FD

- Both AC→D and AD→D have more than one attribute on the left hand side
- Let's check AC→D first

Exercise

- Previous result: S = {A→C, AC→D, AD→B}

2. Check if we can remove any attribute from the left hand side of any FD

- Let's check AC→D first
- Is A redundant?
- If we remove A, then AC→D becomes C→D
- Whether this removal is OK depends on whether C→D is implied by S
- Is C→D implied by S?
- Check: Given S, we have $\{C\}^+ = \{C\}$, which does NOT contain D
- Therefore, C→D is not implied by S, and hence, A is NOT redundant in AC→D

Exercise

- Previous result: S = {A→C, AC→D, AD→B}

2. Check if we can remove any attribute from the left hand side of any FD

- Let's check AC→D first
- Is C redundant?
- If we remove C, then AC→D becomes A→D
- Whether this removal is OK depends on whether A→D is implied by S
- Is A→D implied by S?
- Check: Given S, we have $\{A\}^+$ = {ABCD}, which contain D
- Therefore, A→D is implied by S, and hence, we can simplify AC→D to A→D
- Result: S = {A→C, A→D, AD→B}

Exercise

- Previous result: $S = \{A \rightarrow C, A \rightarrow D, AD \rightarrow B\}$

2. Check if we can remove any attribute from the left hand side of any FD

- Now let's check $AD \rightarrow B$
- Is A redundant?
- If we remove A, then $AD \rightarrow B$ becomes $D \rightarrow B$
- Whether this removal is OK depends on whether $D \rightarrow B$ is implied by S
- Is $D \rightarrow B$ implied by S?
- Check: Given S, we have $\{D\}^+ = \{D\}$, which does NOT contain B
- Therefore, $D \rightarrow B$ is not implied by S, and hence, A is NOT redundant in $AD \rightarrow B$

Exercise

- Previous result: S = {A→C, A→D, AD→B}

2. Check if we can remove any attribute from the left hand side of any FD

- Now let's check AD→B
- Is D redundant?
- If we remove D, then AD→B becomes A→B
- Whether this removal is OK depends on whether A→B is implied by S
- Is A→B implied by S?
- Check: Given S, we have $\{A\}^+$ = {ABCD}, which contain B
- Therefore, A→B is implied by S, and hence, we can simplify AD→B to A→B
- Result: S = {A→C, A→D, A→B}

Exercise

- Previous result: $S = \{A \rightarrow C, A \rightarrow D, A \rightarrow B\}$

3. Remove redundant FDs

- No FD is redundant
- Final minimal basis: $S = \{A \rightarrow C, A \rightarrow D, A \rightarrow B\}$

3NF Decomposition

- Input: A table R with a set of FDs
  1. Find a minimal basis of the FDs
  2. Combine the FDs whose left hand sides are the same
  3. After that, for each FD, construct a table that contains all attributes in the FD
  4. Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  5. Remove subsumed tables
- Example: R(A, B, C, D, E), with BC→DE, A→E, D→A, E→B
  - Minimal basis: BC→D, A→E, D→A, E→B
  - No FDs can be combined
  - Corresponding tables: R_1(B, C, D), R_2(A, E), R_3(A, D), R_4(B, E)
  - Keys of R: AC, BC, CD, CE
  - R_1 contains a key of R

3NF Decomposition

- Input: A
  1. Find
  2. Com
  3. After
     attributes in the FD
  4. Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  5. Remove subsumed tables

> - Why do we need this step?
> - To ensure lossless join decomposition

- Example: R(A, B, C, D, E), with BC→DE, A→E, D→A, E→B
  - Minimal basis: BC→D, A→E, D→A, E→B
  - No FDs can be combined
  - Corresponding tables: R_1(B, C, D), R_2(A, E), R_3(A, D), R_4(B, E)
  - Keys of R: AC, BC, CD, CE
  - R_1 contains a key of R

3NF Decomposition: Adding Key for Lossless Join

- R(A, B, C, D), with $A \rightarrow B$, $C \rightarrow D$
  - Minimal basis: $A \rightarrow B$, $C \rightarrow D$
  - Corresponding tables: $R_1(A, B)$, $R_2(C, D)$
  - Notice that R_1 and R_2 cannot be used to reconstruct R
  - This is why we require the following:
    - Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  - In this case, R has only one key: AC
  - Therefore, we add a table $R_3(A, C)$

3NF Decomposition

- Input: A table R with a set of FDs
  1. Find a minimal basis of the FDs
  2. Combine the FDs whose left hand sides are the same
  3. After that, for each FD, construct a table that contains all attributes in the FD
  4. Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  5. Remove subsumed tables
- We will use an example to explain this step

3NF Decomposition: Remove Subsumed Tables

- Given: A table R, and a set S of FDs
  - e.g., R(A, B, C, D), S = {AB→C, C→A, B→D}
- Step 1: Derive a minimal basis of S
  - The minimal basis of S is {AB→C, C→A, B→D}
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
  - Nothing to be combined
- Step 3: Create a table for each FD remained
  - R_1(A, B, C), R_2(C, A), R3(B, D)
- Step 4: If none of the tables contains a key of the original table R, create a table that contains a key of R (any key would do)
  - R_1(A, B, C) already contains AB, one of the keys of R
- Almost done… But do we need R_2? Everything in R_2 is already in R_1
- Answer: No
- Final decomposition: R_1(A, B, C), R_3(B, D)

3NF Decomposition

- Input: A table R with a set of FDs
  1. Find a minimal basis of the FDs
  2. Combine the FDs whose left hand sides are the same
  3. After that, for each FD, construct a table that contains all attributes in the FD
  4. Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  5. Remove subsumed tables
- In general, we remove a table R’, if all of its attributes are contained in another table R’’

Exercise: 3NF Decomposition

- R(A, B, C, D, E), with A→B, A→C, B→C, E→C, E→D
  1. Find a minimal basis of the FDs
  2. Combine the FDs whose left hand sides are the same
  3. After that, for each FD, construct a table that contains all attributes in the FD
  4. Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  5. Remove subsumed tables

Exercise: 3NF Decomposition

- R(A, B, C, D, E), with $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$, $E \rightarrow C$, $E \rightarrow D$
  - Find a minimal basis
  1. One attribute on the right: $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$, $E \rightarrow C$, $E \rightarrow D$
  2. Remove redundant attributes on the left: $A \rightarrow B$, $A \rightarrow C$, $B \rightarrow C$, $E \rightarrow C$, $E \rightarrow D$
  3. Remove redundant FDs: $A \rightarrow B$, $B \rightarrow C$, $E \rightarrow C$, $E \rightarrow D$

Exercise: 3NF Decomposition

- R(A, B, C, D, E), with A→B, A→C, B→C, E→C, E→D
  1. Minimal basis: A→B, B→C, E→C, E→D
  - Combine the FDs whose left hand sides are the same: A→B, B→C, E→CD
  - For each FD, construct a table that contains all attributes in the FD:

    R_1(A, B), R_2(B, C), R_3(C, D, E)
  - Check if any of the tables contain a key for R; if not, then create a table that contains a key for R:

    Key for R is {AE}, which is not contained in R_1, R_2, or R_3.
  - Create another table R_4(A, E)
  - Remove subsumed tables
  - Final result: R_1(A, B), R_2(B, C), R_3(C, D, E), R_4(A, E)

Exercise 2: 3NF Decomposition

- R(A, B, C, D, E), with A→B, AB→C, C→DE, E→C, E→D
  1. Find a minimal basis of the FDs
  2. Combine the FDs whose left hand sides are the same
  3. After that, for each FD, construct a table that contains all attributes in the FD
  4. Check if any of the tables contain a key for R; if not, then create a table that contains a key for R
  5. Remove subsumed tables

Exercise 2: 3NF Decomposition

- R(A, B, C, D, E), with A→B, AB→C, C→DE, E→C, E→D
  - Find a minimal basis
  1. One attribute on the right: A→B, AB→C, C→D, C→E, E→C, E→D
  2. Remove redundant attributes on the left: A→B, A→C, C→D, C→E, E→C, E→D
  3. Remove redundant FDs: A→B, A→C, C→D, C→E, E→C

Exercise 2: 3NF Decomposition

- R(A, B, C, D, E), with A→B, AB→C, C→DE, E→C, E→D
  - Minimal basis: A→B, A→C, C→D, C→E, E→C
  - Combine the FDs whose left hand sides are the same: A→BC, C→DE, E→C
  - For each FD, construct a table that contains all attributes in the FD:

    R_1(A, B, C), R_2(C, D, E), R_3(C, E)
  - Check if any of the tables contain a key for R; if not, then create a table that contains a key for R:

    Key for R is {A}, which is contained in R_1
  - Remove subsumed tables
  - Final result: R_1(A, B, C), R_2(C, D, E)

Summary

- Poorly designed tables give rise to redundancy, update anomalies, and deletion anomalies
- BCNF eliminates these problems
  - BCNF: For any non-trivial and decomposed FD on a table R, its left hand side is a super-key for R
- But BCNF does not always preserve all FDs
  - We may need to perform a join of multiple tables to check whether an FD holds
- 3NF: slightly weaker than BCNF; has update and deletion anomalies in some rare cases, but preserves all FDs
  - 3NF: For any non-trivial and decomposed FD on a table R, either its left hand side is a super-key for R, or its right hand side is a prime attribute

BCNF or 3NF?

- BCNF is only inferior to 3NF in the sense that sometimes it does not preserve all FDs
- So, go for BCNF if we can find a BCNF decomposition that preserves all FDs
- If such a decomposition cannot be found
  - Go for BCNF if preserving all FDs is not important
  - Go for 3NF otherwise