Third Normal Form (3NF)

→ A relaxation of BCNF that is less strict & allows decompositions that always preserve functional dependencies.

→ Definition: A table satisfies 3NF, if and only if for every non-trivial $X \to Y$

- Either X contains a key (Y depends on key)
- Or each attribute in Y is contained in a key (Y depends on key)

→ Example:

- Given FDs: $C \to B$, $AB \to C$, $BC \to C$
- Keys: {AB}, {AC}
- $AB \to C$ is OK, since AB is a key of R
- $C \to B$ is OK, since B is part of AB
- We ignore $BC \to C$ since it is trival
- So R is in 3NF

→ Example 2:

- Given FDs: $A \to B$, $B \to C$
- Keys: {A}
- $A \to B$ is OK, since is a key of R
- $B \to C$ is not OK, since B doesn't contain a key, and C is not a key of R
- So R is NOT in 3NF

3NF vs BCNF

3NF

→ All attributes (key & non-key attributes) depends (directly or transitively) on candidate keys ↗ multiattribute keys

- But candidate keys may have overlapping attributes
- May result in key-attribute(s) of one key depends on key attribute (s) of another key

→ Example:

- Given FDs: $C \to B$, $AB \to C$, $BC \to C$
- keys: {AB}, {AC}
- key-attributes: A, B, C
- R is in 3NF but $C \to B$ means partial key B depends on partial key C

BCNF

→ In all dependencies (FDs), LHS must contain key (cannot depend on partial key) → LHS cannot have partial key

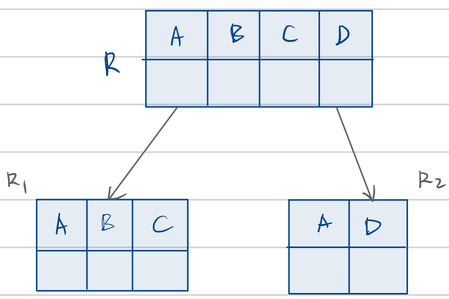
3NF Decomposition

→ Given: A table NOT in 3NF

→ Objective: Decompose it into smaller tables that are in 3NF

→ Example:

- Given: R(A, B, C, D)
- FDs: $AB \to C$, $C \to B$, $A \to D$
- keys: {AB}, {AC}
- R is not in 3NF, due to $A \to D$
- 3NF decomposition of R: $R_1(A, B, C)$, $R_2(A, D)$



3NF Decomposition Algorithm

- → Given: A table R, set S of FDs
  - $R(A, B, C, D)$, $S = \{A \to BD, AB \to C, C \to D, BC \to D\}$
- → Step 1: Derive a minimal basis of S
  - e.g. minimal basis of S is $\{A \to B, A \to C, C \to D\}$
- → Step 2: In the minimal basis, combine the FDs whose LHS are the same
  - e.g. after combining $A \to B$ and $A \to C$, we have $\{A \to BC, C \to D\}$
- → 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 contain a key of the original table R, create a table that contains a key of R
- → Step 5: Remove redundant tables

- → Given: Table $R(A, B, C, D)$, minimal basis $\{A \to B, A \to C, C \to D\}$
- → Step 1: Combine those FDs with the same RHS
  - result: $\{A \to BC, C \to D\}$
- → Step 2: For each FD, create a table that contains all attributes in the FD
  - result: $R_1(A, B, C)$, $R_2(C, D)$
- → Step 3: Remove redundant tables (if any)
- → Tricky issue: Sometimes we also need to add an additional table (see below)

- → Given: Table $R(A, B, C, D)$, minimal basis $\{A \to B, C \to D\}$
- → Step 1: Combine those FDs with the same left hand side
  - result: $\{A \to B, C \to D\}$
- → Step 2: For each FD, create a table that contains all attributes in the FD
  - result: $R_1(A, B)$, $R_2(C, D)$ ⇒ do not ensure lossless join
- → Step 3: If no table contain a key of the original table, add a table that contains a key of the original table
  - Result: $R_1(A, B)$, $R_2(C, D)$, $R_3(A, C)$
- → Step 4: Remove redundant tables (if any)

Example:
- Given R(A, B, C) and {A→B, A→C, B→C, B→A, C→A, C→B}
- Minimal basis 1: {A→B, B→C, C→A}
- Minimal basis 2: {A→C, B→C, C→A, C→B}

Minimal Basis

- preserves FDs; no redundant FDs
- For given set of FDs, MB may not be unique
- Different MB may lead to different 3NF decompositions

- → Minimal Basis of S is the simplified version of S
  - $S = \{A \to BD, AB \to C, C \to D, BC \to D\}$
- → Condition 1: For any FD in the minimal basis, RHS only has 1 attribute
  - $A \to BD$ not in minimal basis
- → Condition 2: No FD in minimal basis is redundant; no FD can be derived from other FDs
  - $BC \to D$ can be derived from $C \to D$
  - ∴ $BC \to D$ not in minimal basis
- → Condition 3: For each FD in the minimal basis, none of the attributes on the LHS is redundant; for any FD in minimal basis, if we remove an attribute from LHS, then resulting FD is a new FD that cannot be derived from original set of FDs
  - remove B from LHS of $AB \to C$, we get $A \to C$
  - $A \to C$ can be derived from S as $\{A\}^+ = \{ABDC\}$
  - $A \to C$ is "hidden" in S
  - Replace $AB \to C$ with $A \to C$ (simpler)
  - ∴ $AB \to C$ not in minimal basis

Algorithm for Minimal Basis Example

→ $S = \{ A \to BD,\ AB \to C,\ C \to D,\ BC \to D \}$

$S: \{ A \to B, \cancel{A \to D}, AB \to C, C \to D, \cancel{BC \to D} \}$

→ Using condition 1 of Minimal Basis,

$S = \{ A \to B,\ A \to D,\ AB \to C,\ C \to D,\ BC \to D \}$ → split only if RHS has 2 attributes

→ Using condition 2 of Minimal Basis,

Is $A \to B$ redundant?

- Without $A \to B$, we have $\{ A \to D, AB \to C, C \to D, BC \to D \}$
- $\{A\}^+ = \{AD\}$ in this case, does not contain B

∴ $A \to B$ not implied by other FDs → not redundant & should not be removed

Is $A \to D$ redundant?

- Without $A \to D$, we have $\{ A \to B, AB \to C, C \to D, BC \to D \}$
- $\{A\}^+ = \{ABCD\}$, which contains D

∴ $A \to D$ implied by other FDs → redundant & should be removed.

Is $AB \to C$ redundant?

- Without $AB \to C$, we have $\{ A \to B, C \to D, BC \to D \}$
- $\{AB\}^+ = \{AB\}$, which does not contain C

∴ $AB \to C$ is not implied by other FDs → not redundant & should not be removed

Is $C \to D$ redundant?

- Without $C \to D$, we have $\{ A \to B, AB \to C, BC \to D \}$
- $\{C\}^+ = \{C\}$, which does not contain D

∴ $C \to D$ is not implied by other FDs → not redundant & should not be removed

Is $BC \to D$ redundant?

- Without $BC \to D$, we have $\{ A \to B, AB \to C, C \to D \}$
- $\{BC\}^+ = \{BCD\}$, which contains D

∴ $BC \to D$ is implied by other FDs, hence is redundant & should be removed.

→ Using condition 3 of minimal basis,

$S = \{ A \to B, AB \to C, C \to D \}$

Only $AB \to C$ has >1 attribute on LHS

Is A redundant?

- removing A → $AB \to C$ becomes $B \to C$
- $\{B\}^+ = \{B\}$, which does not contain C

∴ $B \to C$ is not "hidden" in S, hence A is not redundant

Is B redundant?

- removing B → $AB \to C$ becomes $A \to C$
- $\{A\}^+ = \{ABCD\}$, which contains C

∴ $A \to$ is "hidden" in S, hence B is redundant

Final minimal basis: $S = \{ A \to B,\ A \to C,\ C \to D \}$

BCNF Vs 3NF

- BCNF: For any non-trivial FD → can also be a key
  - its left hand side (lhs) is a superkey
- 3NF: For any non-trivial FD
  - Either its lhs is a superkey
  - Or each attribute on its right hand side appear in a key
- Observation: BCNF is stricter than 3NF
- Therefore
  - A table that satisfies BCNF must satisfy 3NF, but not vice versa
  - A table that violates 3NF must violate BCNF, but not vice versa

- BCNF Decomposition:
  - Avoids insertion, deletion, and update anomalies
  - Eliminates most redundancies
  - But does not always preserve all FDs
- 3NF Decomposition:
  - Avoids insertion, deletion, and update anomalies
  - May lead to a bit more redundancy than BCNF
  - Always preserve all FDs
- So which one to use?
- A logical approach
  - Go for a BCNF decomposition first
  - If it preserves all FDs, then we are done
  - If not, then go for a 3NF decomposition instead

When an FD is not violating, a table is NOT formed

Violating or non-violating FDs in 3NF will become a table
↓
may have attributes repeated in >1 table
↓
perserves all FDs

Extra knowledge :

First Normal Form (1NF)

- Key-attribute: An attribute in a multi-attribute key
  - ABC is key, then A, B, or C is key attribute
  - Key-attribute(s) = Partial key or part of a key; AB is partial key
- 1NF: All attributes have atomic values
- 2NF: Every non-key attribute is dependent on the whole of every candidate key
  - But, may still have (non-key-attribute X) → (non-key-attribute Y) in the relation

Second Normal Form (2NF)

- 2NF: Every non-key attribute is dependent on the whole of **EVERY** candidate key
  - But, may still have (non-key-attribute X) → (non-key-attribute Y) in the relation
- Example:
  - Given FDs: A→B, A→C, B→C
  - Key: {A}
  - Non-key attributes: B, C
  - A→B means B depends on key
  - A→C means C depends on key
  - But B→C means non-key C depends on non-key B
  - So R is in 2NF

R

A	B	C