BCNF Decomposition and Candidate Key Analysis

Consider the relation schema R(ABCDEFG) with the following functional dependencies:
A -> C
B -> A
CD -> B
E -> F
G -> E

i. Determine if "CDG" and "BCE" are candidate keys. Justify your answer.
ii. Decompose the schema into BCNF and explain the decomposition process

Part i: Determine if "CDG" and "BCE" are Candidate Keys

We must compute closures of **CDG** and **BCE** to check whether they are **superkeys** (i.e., their closure contains **all attributes in R**) and **minimal** (i.e., no subset of them is also a superkey).

1. Check if CDG is a candidate key

We compute **$(CDG)^+$**

Start: CDG → {C, D, G}

- CD → B ⇒ add B
- B → A ⇒ add A
- A → C ⇒ already there
- A → C ⇒ OK
- G → E ⇒ add E
- E → F ⇒ add F

So $(CDG)^+$ = {C, D, G, B, A, E, F}
→ Contains: A, B, C, D, E, F, G
So **CDG is a superkey**

Now check if it's minimal:

- Try removing **C**:
  - $(DG)^+$ = {D, G}
    - G → E → F, nothing else
    - No CD → B → A etc.
      ⇒ Doesn't work

- Try removing **D**:
  - $(CG)^+ = \{C, G\}$
    - G → E → F
    - CD → B: CD missing
      ⇒ Doesn't work
- Try removing **G**:
  - $(CD)^+ = \{C, D\}$
    - CD → B → A → C
    - But G → E → F missing
      ⇒ Doesn't work

CDG is minimal → So **CDG is a candidate key**

2. Check if BCE is a candidate key

Compute **$(BCE)^+$**

Start: BCE → {B, C, E}

- B → A ⇒ add A
- A → C ⇒ already present
- E → F ⇒ add F
- CD → B — need D and C
- G → E — G not present

Still missing D, G → So try getting D
We don't have any FD giving us D or G

So $(BCE)^+ = \{B, C, E, A, F\}$
→ Missing D, G ⇒ **Not a superkey**

So **BCE is not a candidate key**

Answer for (i):

- **CDG is a candidate key** (justified by closure)
- **BCE is not a candidate key**

Part ii: BCNF Decomposition

Step 1: Recall BCNF Rule:

A relation is in **BCNF** if for **every non-trivial functional dependency** X → Y, **X is a superkey** of the relation.
So, for each FD, check:
Is the LHS a superkey? ➔ If not, decompose

A relation is in **BCNF** if for every FD $X \rightarrow Y$, **X is a superkey**.

Step 1: Find violations of BCNF

A relation violates **BCNF** if there is a non-trivial functional dependency **X → Y** where **X is not a superkey**.

FD	Is X a superkey?	Violation?
A → C	A^+ = {A, C} → ✕	Yes
B → A	B^+ = {B, A, C} → ✕	Yes
CD → B	CD^+ = {C, D, B, A, C} → Partial → ✕	Yes
E → F	E^+ = {E, F} → ✕	Yes
G → E	G^+ = {G, E, F} → ✕	Yes

All FDs violate BCNF ⇒ Decompose repeatedly

We'll check each FD:

1. A →C

A^+ = {A, C} → Not all attributes
⇒ A is **not a superkey** ⇒ **violates BCNF**
→ Decompose on A → C

New relations:

- R_1(A, C)
- R_2(A, B, D, E, F, G)

Now decompose R_2(A, B, D, E, F, G)

FDs in R_2:

- B → A
- CD → B → not in R_2 (no C)
- E → F
- G → E

Let's test **B → A** in R_2

- B^+ = {B, A}
  → Not all attributes ⇒ violates BCNF

Decompose R_2 on **B → A**

New relations:

- R_3(B, A)
- R_4(B, D, E, F, G)

Now decompose R_4(B, D, E, F, G)

FDs left: E → F, G → E

Check **E → F**

When a relation Ri violates BCNF:
decompose **Ri** using the violating **functional dependency**:
α → β
(where α is not a superkey and the dependency is non-trivial)

BCNF Decomposition Rule:
You create two new relations:
i. $Ri_1 = \alpha \cup \beta$
This relation **captures** the functional dependency.
It includes the **left-hand side (α)** and **right-hand side (β)** of the FD.
ii. $Ri_2 = Ri - (\beta - \alpha)$
This is the **remaining attributes** of the original Ri, **excluding** the attributes in β **that are not already in α**.
It retains α (because it's shared between Ri_1 and Ri_2) and any other attributes that weren't part of β.

Why this works:
Ri_1 preserves the FD α → β
Ri_2 keeps the rest of the schema but still includes α (needed for a **lossless join**)
Together, they let you **reconstruct the original Ri** using a **natural join**

Example
Let's say we have a relation:
Ri = (A, B, C, D)
Functional Dependency: A → B
α = A
β = B
Now apply the BCNF decomposition:
i. $Ri_1 = \alpha \cup \beta = A \cup B$ = **(A, B)**
→ This holds the dependency A → B.
ii. $Ri_2 = Ri - (\beta - \alpha)$
→ β − α = B − A = {B}
→ Ri − {B} = (A, B, C, D) − {B} = **(A, C, D)**
→ So **Ri_2 = (A, C, D)**
Now we have:
Ri_1 = (A, B) — FD: A → B
Ri_2 = (A, C, D) — remaining attributes with A retained for lossless join

Intuition:
We split off the part of the relation where the **FD holds cleanly (Ri_1)**
The **rest of the relation (Ri_2)** keeps everything else but ensures **α (A)** is still there so we can **join it back**

- $E^+ = \{E, F\}$
  ⇒ Not a superkey ⇒ violates BCNF

Decompose on $E \rightarrow F$

New relations:

- $R_5(E, F)$
- $R_6(B, D, E, G)$

Now check $R_6(B, D, E, G)$

Only relevant FD: $G \rightarrow E$

- $G^+ = \{G, E\}$
  → Not all attributes ⇒ violates BCNF

Decompose on $G \rightarrow E$

Final relations:

- $R_7(G, E)$
- $R_8(B, D, G)$

Final BCNF Decomposition:

1. **R_1(A, C)** from $A \rightarrow C$
2. **R_3(B, A)** from $B \rightarrow A$
3. **R_5(E, F)** from $E \rightarrow F$
4. **R_7(G, E)** from $G \rightarrow E$
5. **R_8(B, D, G)** — no violating FD ⇒ BCNF

Final Answer :

(i) Candidate Keys:

- **CDG is a candidate key**
- **BCE is not a candidate key**

(ii) BCNF Decomposition:

The relation **R(ABCDEFG)** is decomposed into:
R_1(A, C)
R_3(B, A)
R_5(E, F)
R_7(G, E)
R_8(B, D, G)

Definition of BCNF

A relation schema **R** is in **BCNF** if **for every non-trivial functional dependency $\alpha \to \beta$**, one of the following must be true:

- **$\alpha \to \beta$ is trivial**, i.e., $\beta \subseteq \alpha$
- **α is a superkey** of R

2. How to Check BCNF

To test if a functional dependency **violates BCNF**:

- Compute **α^+ (closure of α)**
- If **α^+ contains all attributes in R**, then α is a superkey ⇒ BCNF holds
- Otherwise, the dependency **violates BCNF**

Note: It's okay to only check dependencies in F, *not* F^+, when checking the **original relation**, but **not enough** when testing **a decomposed relation**.
For decompositions, you **must check F^+** as **hidden dependencies** (e.g., $A \to C$ from $A \to B$, $B \to C$) can exist.

3. Example :

R = (A, B, C), F = {A → B, B → C}

- **Candidate key**: A
- But $B \to C$ violates BCNF because B is **not** a superkey
- So, decompose:
  - R1 = (A, B) with $A \to B$ → **BCNF**
  - R2 = (B, C) with $B \to C$ → **BCNF**

This decomposition is:

- Lossless-join
- Dependency preserving

4. Important Caution in BCNF Decomposition

However, using only F is incorrect when testing a relation in a decomposition of R...

This is **important**.

Example:

- R = (A, B, C, D), F = {$A \to B$, $B \to C$}

- F^+ includes $A \rightarrow C$
- After decomposition:
  - R1 = (A, B) — check using F, fine
  - R2 = (A, C, D) — none of F's FDs include only A, C, D
    You **might think** R2 is in BCNF
    But from F^+, we know $\mathbf{A \rightarrow C}$ exists ⇒ **violates BCNF**

Thus, $\mathbf{F^+}$ must be used when testing BCNF **after decomposition**.

5. BCNF Decomposition Algorithm

1. Start with result = {R}
2. While there is a relation Ri in result not in BCNF:
   a. Find a dependency $\alpha \rightarrow \beta$ in Ri that violates BCNF
   b. Replace Ri with two schemas:
      i. Ri1 = $\alpha \cup \beta$
      ii. Ri2 = Ri - (β - α)
3. Repeat until all Ri are in BCNF

Guarantees:

- Lossless-join
- But **not always dependency preserving**

6. Example: Dependency Not Preserved in BCNF

- R = (J, K, L), F = {JK → L, L → K}
- Candidate keys = JK, JL
- R is **not in BCNF**
- Any decomposition will lose the dependency JK → L

So: **Not all BCNF decompositions preserve all functional dependencies**