Decomposition of a Relation Schema:

- If a relation is not in a desired normal form, it can be decomposed into multiple relations that each are in that normal form.
- Suppose that relation R contains attributes A1 ... An. A decomposition of R consists of replacing R by two or more relations such that:
    - Each new relation scheme contains a subset of the attributes of R, and
    - Every attribute of R appears as an attribute of at least one of the new relations.
- When we decompose a relation schema R with a set of functional dependencies F into R1, R2, …, Rn we want:
    - Lossless-join decomposition
    - No redundancy
    - Dependency preservation
- To test if we have lossless-joins, we can use the **chase test**.

Chase Test:

- **E.g.** Given R(K, L, M, N, P), the FDs
  L → P
  MP → K
  KM → P
  LM → N
  and the fact that we decomposed R into
  R1(K, L, M)
  R2(L, M, N)
  R3(K, M, P)
  Use the chase test to see if the decomposition is lossless.

  Soln:
  First, we make a table with the attributes listed on the first row and the decomposed relations listed on the first column. Then, for each decomposed relation, we put a α under the attributes that the relation has. I.e. If relation Rn has attribute x, we put a α in the cell (Rn, x).

	K	L	M	N	P
R1	α	α	α		
R2	α	α		α	
R3	α		α		α

  Second, we run each FD through all the relations and if we can get a new attribute, we put a α under the new attribute we got for that relation.

Let’s start with the FD L → P. Since R1 and R2 both have attribute L but not attribute P, they both get a new attribute. Hence, I’ll put a α in the cells (R1, P) and (R2, P).

	K	L	M	N	P
R1	α	α	α		α
R2	α	α		α	α
R3	α		α		α

We go to the next FD, MP → K. Nothing changes since R1, R2 and R3 all have α under the K column.

We go to the next FD, KM → P. Nothing changes since R1, R2 and R3 all have α under the P column.

We go to the next FD, LM → N. We add a α to (R1, N).

	K	L	M	N	P
R1	α	α	α	α	α
R2	α	α		α	α
R3	α		α		α

We stop here since R1 has α for each of the attributes. This means that the decomposition is lossless.
Note: If we didn’t have a relation with α’s under all the attributes, we’d repeat the process with the FDs until we get a relation with α’s under all the attributes or until we can’t change anything anymore.

- **E.g.** Given R(A, B, C, D) and the FDs
  A→B
  B→C
  CD→A
  and the fact that we decomposed R into
  R1 = {A, D}
  R2 = {A, C}
  R3 = {B, C, D}.
  Use the chase test to see if the decomposition is lossless.

Soln:
This is the initial table.

	A	B	C	D
R1	α			α
R2	α		α	
R3		α	α	α

Using the FD A → B, we get the new table

	A	B	C	D
R1	α	α		α
R2	α	α	α	
R3		α	α	α

Using the FD A → C, we get the new table

	A	B	C	D
R1	α	α	α	α
R2	α	α	α	
R3		α	α	α

We stop here because we see that R1 has α for each attribute. This means that the decomposition is lossless.

<u>Decomposition for 1NF:</u>

- A table is in 1NF if each cell can only contain 1 value.
- **Decomposition Method:**
    - If a cell contains multiple values, create a new row for each value.

<u>Decomposition for 2NF:</u>

- A table is in 2NF if:
    - It is in 1NF.
    - It does not have any partial dependencies (pds).
- **Decomposition Method:**
    - Identify all the candidate keys.
    - Identify the prime and non-prime attributes.
    - Identify the partial dependencies.
    - Decompose the relation for the candidate keys and all pds.
- **E.g.** Given R(A, B, C, D, E) and the FDs
  AB → C
  D → E
  Determine if R is in 2NF, and if it isn't decompose it so that the relations are in 2NF.

  Soln:
  Notice how the RHS of the FDs do not contain A, B and D. Hence, we know that all candidate keys must contain at least A, B and D. The closure of ABD is {A, B, C, D, E}, so in this case, the candidate key is {A, B, D}.

  Since the candidate key is {A, B, D}, the prime attributes are A, B and D and the non-prime attributes are C and E.

  We see that AB → C and D → E are pds.

I will decompose R into:
R1(A, B, C)
R2(D, E)
R3(A, B, D)

- **E.g.** Given R(A, B, C, D) and the FDs
  AB → CD
  A → D
  Determine if R is in 2NF, and if it isn't decompose it so that the relations are in 2NF.

  Soln:
  Notice how the RHS of the FDs do not contain A and B. Hence, we know that all candidate keys must contain at least A and B. The closure of AB is {A, B, C, D}, so in this case, the candidate key is {A, B}.

  Since the candidate key is {A, B}, the prime attributes are A and B and the non-prime attributes are C and D.

  We see that A → D is a pd.
  AB → CD isn't a pd since CD depends on A and B.

  I will decompose R into:
  R1(A, D)
  R2(A, B, C)

- **E.g.** Given R(A, B, C, D, E) and the FDs
  AB → C
  B → D
  E → D
  Determine if R is in 2NF, and if it isn't decompose it so that the relations are in 2NF.

  Soln:
  Notice how the RHS of the FDs do not contain A, B and E. Hence, we know that all candidate keys must contain at least A, B and E. The closure of ABE is {A, B, C, D, E}, so in this case, the candidate key is {A, B, E}.

  Since the candidate key is {A, B, E}, the prime attributes are A, B and E and the non-prime attributes are C and D.

  AB → C, B → D and E → D are all pds.

  I will decompose R into:
  R1(A, B, C)
  R2(B, D)
  R3(E, D)
  R4(A, B, E)

<u>Decomposition for 3NF:</u>

- A table is in 3NF if:
    - It is in 2NF.
    - It does not have any transitive dependencies (tds).
- **Decomposition Method:**
    - Identify all the candidate keys.
    - Identify the prime and non-prime attributes.
    - Identify the partial dependencies and the transitive dependencies.
    - Decompose the relation for the candidate keys, all pds and all tds.
- **E.g.** Given R(A, B, C, D, E, F, G, H) and the FDs
  A→B
  ABCD→E
  EF→GH
  ACDF→EG
  Determine if R is in 3NF, and if it isn't decompose it so that the relations are in 3NF.

  Soln:
  We see that the RHS of any FD does not include A, C, D and F. Hence, any candidate keys must contain at least A, C, D and F. The closure of ACDF is {A, B, C, D, E, F, G, H}. Hence, {A, C, D, F} is a candidate key.

  The prime attributes are A, C, D, F.
  The non-prime attributes are B, E, G, H.

  A → B is a pd.
  ABCD → E is a td.
  EF → GH is a td.

  I will decompose R into
  R1(A, B)
  R2(A, C, D, E)
  R3(E, F, G)
  R4(E, F, H)
  R5(A, C, D, F)

- **E.g.** Given R(A, B, C, D) and the FDs
  C → DA
  B → C
  Determine if R is in 3NF, and if it isn't decompose it so that the relations are in 3NF.

  Soln:
  We see that B is not in the RHS of any FD, so all candidate keys must contain at least B. The closure of B is {B, C, D, A}, so {B} is a candidate key.

  The prime attribute is B.
  The non-prime attributes are A, C, D.

  We see that C → DA is a td.

I will decompose R into:
R1(B, C)
R2(C, D, A)

- **E.g.** Given R(A, B, C, D) and the FDs
  AB → CD
  C → A
  D → B
  Determine if R is in 3NF, and if it isn't decompose it so that the relations are in 3NF.

  Soln:
  We see that the candidate keys are {A, B}, {C, D}, {A, D} and {B, C}.
  The prime attributes are A, B, C, and D.
  There are no non-prime attributes.
  Since all attributes are prime attributes, we will never get P → NP (pd) or NP → NP (td).
  Hence, R is in 3NF already.

Decomposition for BCNF:

- A table is in BCNF if:
  - It is in 3NF.
  - For each non-trivial FD X → Y, X must be a super key.
- **Decomposition Method #1:**
  decompose (R, X → Y):
  R1(R - Y)
  R2(X + Y)
  Project FDs onto R1 and R2 recursively call decompose on R1 and R2 for BCNF violations.
- **Decomposition Method #2:**
  decompose (R, X → Y):
  R1(X^+)
  R2(R - (X^+ - X))
  Project FDs onto R1 and R2 recursively call decompose on R1 and R2 for BCNF violations.
- **E.g.** Given R(A, B, C) and the FDs
  A → B
  B → C

  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  The candidate key is {A}.
  Hence, B → C is a td, meaning that R is not in BCNF.
  I'll decompose R into
  R1(A, B)
  R2(B, C)

- **E.g.** Given R(A, B, C, D, E) and the FDs
  A → B
  BC → D

  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  The candidate key is {A, C, E}.
  Hence, A → B is a pd and BC → D is a td, meaning that R is not in BCNF.
  I will use A → B.
  I'll decompose R into
  R1(A, B) ← R1(X^+)
  R2(A, C, D, E) ← R2(R - (X^+ - X))

  We can decompose R2 further.
  We know that since A → B and BC → D, by the pseudo transitivity axiom, AC → D.
  This means that R2 isn't in BCNF as AC isn't a super key, so we have to decompose R2 further.
  R21(A, C, D)
  R22(A, C, E)

  The final decomposition is
  R1(A, B)
  R21(A, C, D)
  R22(A, C, E)

- **E.g.** Given R(A, B, C, D, E, H) and the FDs
  A → BC
  E → HA

  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  The candidate key is {D, E}.
  We see that A → BC is td and E → HA is pd. Hence, R is not in BCNF.
  I will use A → BC.
  I will decompose R into
  R1(A, B, C) ← R1(X^+)
  R2(A, D, E, H) ← R2(R - (X^+ - X))

  We see that there's an issue with R2.
  We still have the FD E → HA but E isn't a super key.
  We have to decompose R further.
  R21(D, E)
  R22(E, H, A)

  The final decomposition is
  R1(A, B, C)
  R21(D, E)
  R22(E, H, A)

- **E.g.** Given R(A, B, C, D) and the FDs
  C → DA
  B → C
  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  We see that B is not in the RHS of any FD, so all candidate keys must contain at least B.
  The closure of B is {B, C, D, A}, so {B} is a candidate key.

  The prime attribute is B.
  The non-prime attributes are A, C, D.

  We see that C → DA is a td.

  I will decompose R into
  R1(C, D, A)
  R2(B, C)

- **E.g.** Given R(A, B, C, D) and the FDs
  AB → CD
  C → A
  D → B
  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  We see that the candidate keys are {A, B}, {C, D}, {A, D} and {B, C}.
  We see that C → A and D → B violates BCNF.
  I will use C → A.
  I will decompose R into
  R1(C, A)
  R2(B, C, D)

  We see that the FD D → B still applies for R2, so it is not in BCNF.
  I will decompose R2 into
  R21(D, B)
  R22(C, D)

  The final decomposition is
  R1(C, A)
  R21(D, B)
  R22(C, D)

- **E.g.** Given R(A, B, C, D, E) and the FDs
  A → BC
  C → DE
  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  We see that the candidate key is {A}.
  Hence, C → DE is a td, meaning that R is not in BCNF.
  I will decompose R into
  R1(C, D, E)
  R2(A, B, C)

- **E.g.** Given R(A, B, C, D) and the FDs
  AB → C
  B → D
  C → A
  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  We see that the candidate keys are {A, B} and {B, C}.
  Hence, B → D and C → A violate BCNF as the LHS are not super keys.
  I will use B → D.
  I will decompose R into
  R1(B, D)
  R2(A, B, C)

  We see that the FD C → A still holds for R2, so it is not in BCNF.
  I will decompose R2 into
  R21(C, A)
  R22(B, C)

  The final decomposition is
  R1(B, D)
  R21(C, A)
  R22(B, C)

- **E.g.** Given R(A, B, C, D) and the FDs
  A → BCD
  BC → AD
  D → B
  Determine if R is in BCNF, and if it isn't decompose it so that the relations are in BCNF.

  Soln:
  We see that the candidate keys are {A} and {B, C}.
  Hence, D → B is a td and violates BCNF.
  I will decompose R into
  R1(D, B)
  R2(A, C, D)

<u>Decomposition for 4NF:</u>

- A table is in 4NF if:
  - It is in BCNF.
  - It does not have any multi-valued dependencies (mvds).
- **Decomposition Method:**
  decompose (R, X -->> Y):
  R1 = XY
  R2 = X union (R - Y)
  Repeat on R1 and R2 until all relations are in 4NF.
- **E.g.** Given R(A, B, C, D) and the FD
  $A \rightarrow B$
  and the mvds
  A -->> C
  A -->> D
  Determine if R is in 4NF, and if it isn't decompose it so that the relations are in 4NF.

  Soln:
  We see that the candidate key is {A, C, D}.
  We see that $A \rightarrow B$ violates BCNF while A -->> C and A -->> D violates 4NF.
  I will use $A \rightarrow B$.
  I will decompose R into
  R1(A, B)
  R2(A, C, D)

  R2 violates 4NF because of the mvds.
  I will decompose R2 using A -->> C.
  R21(A, C)
  R22(A, D)

  The final decomposition is
  R1(A, B)
  R21(A, C)
  R22(A, D)