Decomposition, 3NF, BCNF

Dr. Bassam Hammo
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 $A_1 \ldots A_n$. 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.
Normalization Using Functional Dependencies

- When we decompose a relation schema $R$ with a set of functional dependencies $F$ into $R_1, R_2, \ldots, R_n$ we want
  - **Lossless-join Decomposition** (complete reproduction)
  - **No Redundancy** (BCNF or 3NF)
  - **Dependency Preservation**
Lossless-join Decomposition

- All attributes of an original schema \((R)\) must appear in the decomposition \((R_1, R_2)\):
  \[ R = R_1 \cup R_2 \]
- For all possible relations \(R_i\) on schema \(R\)
  \[ R = \prod_{R_1} (R) \bowtie \prod_{R_2} (R) \]
- We want to be able to reconstruct big (e.g. universal) relation by joining smaller ones (using natural joins)
  (i.e. \(R_1 \bowtie R_2 = R\))
Example (Lossless-Join)

A B C
a_1 b_1 c_1
a_1 b_2 c_2

A C
a_1 c_1
a_1 c_2

B C
b_1 c_1
b_2 c_2

A B C
a_1 b_1 c_1
a_1 b_2 c_2
Example (Lossy-Join)

\[
\begin{array}{c|c|c}
A & B & C \\
\hline
a_1 & b_1 & c_1 \\
\hline
a_1 & b_2 & c_2 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
A & B & C \\
\hline
a_1 & b_1 & c_1 \\
\hline
a_1 & b_1 & c_2 \\
\hline
a_1 & b_2 & c_1 \\
\hline
a_1 & b_2 & c_2 \\
\end{array}
\]

decompose

join
Testing for Lossless-Join Decomposition

• Rule: A decomposition of $R$ into $(R_1, R_2)$ is lossless, iff:

\[
R_1 \cap R_2 \rightarrow R_1 \quad \text{or} \quad R_1 \cap R_2 \rightarrow R_2
\]

in $F^+$. 
Exercise: Lossless-join Decomposition

\[ R = \{ A, B, C, D, E \} . \]
\[ F = \{ A \rightarrow BC, \; CD \rightarrow E, \; B \rightarrow D, \; E \rightarrow A \} . \]

Is the following decomposition a lossless join?

1. \( R_1 = \{ A, B, C \}, \; R_2 = \{ A, D, E \} \)
   Since \( R_1 \cap R_2 = A \), and A is a key for \( R_1 \),
   the decomposition is lossless join.

2. \( R_1 = \{ A, B, C \}, \; R_2 = \{ C, D, E \} \)
   Since \( R_1 \cap R_2 = C \), and C is not a key for \( R_1 \) or \( R_2 \),
   the decomposition is not lossless join.
Dependency Preserving Decomposition

- The decomposition of a relation scheme $R$ with FDs $F$ is a set of tables (fragments) $R_i$ with FDs $F_i$.
- $F_i$ is the subset of dependencies in $F^+$ (the closure of $F$) that include only attributes in $R_i$.
- The decomposition is **dependency preserving** iff
  \[(\bigcup_{i} F_i)^+ = F^+\]
- In other words: we want to minimize the cost of global integrity constraints based on FD’s (i.e. avoid big joins in assertions)
  \[(F_1 \cup F_2 \cup ... \cup F_n)^+ = F^+\]
Exercise: Non-Dependency Preserving Decomposition

\[ R = (A, B, C), \ F = \{ A \rightarrow B, B \rightarrow C, A \rightarrow C \} \]

Key: A

Assume there is a dependency \( B \rightarrow C \), where the LHS is not the key, meaning that there can be considerable redundancy in \( R \).

Solution: Break it in two tables \( R1(A,B) \), \( R2(A,C) \)
The decomposition is **lossless** because the common attribute \( A \) is a key for \( R_1 \) (and \( R_2 \))

The decomposition is not dependency preserving because:

\[ F_1 = \{ A \rightarrow B \}, \]
\[ F_2 = \{ A \rightarrow C \} \text{ and } (F_1 \cup F_2)^+ \neq F^+ \]

But, we lost the FD \( \{ B \rightarrow C \} \)

- In practical terms, each FD is implemented as a constraint or assertion, which it is checked when there are updates. In the above example, in order to find violations, we have to join \( R_1 \) and \( R_2 \). Which can be very expensive.
Exercise: Dependency Preserving Decomposition

\( R = (A, B, C), \ F = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\} \)  Key: A

Solution: Break it in two tables \( R_1(A, B), R_2(B, C) \)

- The decomposition is **lossless** because the common attribute \( B \) is a key for \( R_2 \)
- The decomposition is **dependency preserving** because \( F_1 = \{A \rightarrow B\}, F_2 = \{B \rightarrow C\} \) and \((F_1 \cup F_2)^+ = F^+\)
- Violations can be found by inspecting the individual tables, without performing a join.
- What about \( A \rightarrow C \) ?

  If we can check \( A \rightarrow B \), and \( B \rightarrow C, A \rightarrow C \) is implied.
Exercise 2 : FD-Preserving Decomposition

\( R = \{A,B,C,D,E\} \).
\( F = \{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, E \rightarrow A \} \)
\( R_1 = \{A,B,C\}, \quad R_2 = \{A,D,E\} \)

Is the above decomposition dependency-preserving?

No.

\( CD \rightarrow E \) and \( B \rightarrow D \) are lost.
3NF

Third Normal Form
Decomposition
Third Normal Form

3NF: A schema $R$ is in third normal form (3NF) if

for all FD $\alpha \rightarrow \beta$ in $F^+$, at least one of the following holds:

1. $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$).
2. $\alpha$ is a superkey for $R$.
3. Each attribute $A$ in $\beta - \alpha$ is contained in a candidate key for $R$ (prime).

- The decomposition is both lossless-join and dependency-preserving
Third Normal Form

- A relational schema $R$ is in 3NF if for every FD $X \rightarrow A$ associated with $R$ either:
  - $A \subseteq X$ (i.e., the FD is trivial) or
  - $X$ is a superkey of $R$ or
  - $A$ is part of some key (not just superkey!)
- 3NF weaker than BCNF (every schema that is in BCNF is also in 3NF)
Third Normal Form

- Compromise - Not all redundancy removed, but dependency-preserving decompositions are always possible
- 3NF decomposition is based on the concept of *minimal cover of a set of FDs*
Decomposition into 3NF

- **Decomposition**
  - Given: relation R, set F of functional dependencies
  - Find: decomposition of R into a set of 3NF relation R_i
  - Algorithm:

  1. Eliminate redundant FDs, resulting in a canonical cover Fc of F
  2. Create a relation R_i = XY for each FD X → Y in Fc
  3. If the key K of R does not occur in any relation R_i, create one more relation R_i = K
Computing Minimal Cover

- **step 1**: RHS of each FD is a single attribute.
- **step 2**: Eliminate unnecessary attributes from LHS.
  - Algorithm: If FD $XB \rightarrow A \in T$ (where $B$ and $A$ are single attributes) and $X \rightarrow A$ is entailed by $T$, then $B$ was unnecessary.
- **step 3**: Delete unnecessary FDs from $T$
  - Algorithm: If $T - \{f\}$ entails $f$, then $f$ is unnecessary.
    - If $f$ is $X \rightarrow A$ then check if $A \in X^+_{T - \{f\}}$
Example

• \{A \rightarrow B, ABCD \rightarrow E, EF \rightarrow GH, ACDF \rightarrow EG\}
• Make RHS a single attribute: \{A \rightarrow B, ABCD \rightarrow E, EF \rightarrow G, EF \rightarrow H, ACDF \rightarrow E, ACDF \rightarrow G\}
• Minimize LHS: ACD \rightarrow E instead of ABCD \rightarrow E
• Eliminate redundant FDs
  • Can ACDF \rightarrow G be removed?
  • Can ACDF \rightarrow E be removed?
• Final answer: \{A \rightarrow B, ACD \rightarrow E, EF \rightarrow G, EF \rightarrow H\}
Example

- Relation: $R=\text{CSDJPQV}$
- FDs: $C \rightarrow \text{CSDJPQV}$, $SD \rightarrow P$, $JP \rightarrow C, J \rightarrow S$
- Find minimal cover: \{ $C \rightarrow J$, $C \rightarrow D$, $C \rightarrow Q$, $C \rightarrow V$, $JP \rightarrow C$, $J \rightarrow S$, $SD \rightarrow P$ \}
- Combine LHS: \{ $C \rightarrow JDQV$, $JP \rightarrow C$, $J \rightarrow S$, $SD \rightarrow P$ \}
- New relations: $\text{CJDQV}$, $\text{JPC}$, $\text{JS}$, $\text{SDP}$
- Since $\text{CJDQV}$ is a superkey we are done!
BCNF

BCNF Normal Form Decomposition
Boyce-Codd Normal Form

- **BCNF:** A schema $R$ is in BCNF with respect to a set $F$ of functional dependencies, if for all functional dependencies in
  - $F^+$ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:
    1. $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
    2. $\alpha$ is a superkey for $R$

- In other words, the left part of any non-trivial dependency must be a superkey.
- If we do not have redundancy in $F$, then for each $\alpha \rightarrow \beta$, $\alpha$ must be a candidate key.
- **The decomposition is lossless-join but may not be dependency-preserving**
Decomposing into BCNF Schemas

- For all dependencies $A \rightarrow B$ in $F+$, check if $A$ is a superkey
  - By using attribute closure

- If not, then
  - Choose a dependency in $F+$ that breaks the BCNF rules, say $A \rightarrow B$
  - Create $R1 = A \cdot B$
  - Create $R2 = A \cdot (R - B - A)$
  - Note that: $R1 \cap R2 = A$ and $A \rightarrow AB (= R1)$, so this is lossless decomposition

- Repeat for $R1$, and $R2$
  - By defining $F1+$ to be all dependencies in $F$ that contain only attributes in $R1$
  - Similarly $F2+$
BCNF Decomposition

• Suppose $R = (R; F)$ is not in BCNF

• In general: Let $X \rightarrow Y \in F$ be a violating FD
  • Decompose into $XY$ and $(R - Y) \cup X$

If either $R-A$ or $XA$ is not in BCNF, decompose them further recursively
BCNF Example #1

R = (A, B, C)
F = {A → B, B → C}
Candidate keys = {A}

B → C

R1 = (B, C)
F1 = {B → C}
Candidate keys = {B}
BCNF? = true

R2 = (A, B)
F2 = {A → B}
Candidate keys = {A}
BCNF? = true
BCNF Example #2

$$R = (A, B, C, D, E)$$

$$F = \{A \rightarrow B, BC \rightarrow D\}$$

Candidate keys = \{ACE\}

BCNF = Violated by \{A \rightarrow B, BC \rightarrow D\} etc...

From A \rightarrow B and BC \rightarrow D by pseudo-transitivity

- A \rightarrow B
  - R1 = (A, B)
  - F1 = \{A \rightarrow B\}
  - Candidate keys = \{A\}
  - BCNF = true

- AC \rightarrow D
  - R2 = (A, C, D, E)
  - F2 = \{AC \rightarrow D\}
  - Candidate keys = \{ACE\}
  - BCNF = false (AC \rightarrow D)

Dependency preservation ???
We can check:
  - A \rightarrow B (R1), AC \rightarrow D (R3),
  - but we lost BC \rightarrow D
So this is not a dependency-preserving decomposition
Example #3

\[ R = (A, B, C, D, E) \]
\[ F = \{ A \rightarrow B, BC \rightarrow D \} \]
Candidate keys = \{ACE\}
BCNF = Violated by \{A \rightarrow B, BC \rightarrow D\} etc…

\[ BC \rightarrow D \]

R1 = (B, C, D)
F1 = \{BC \rightarrow D\}
Candidate keys = \{BC\}
BCNF = true

R2 = (B, C, A, E)
F2 = \{A \rightarrow B\}
Candidate keys = \{ACE\}
BCNF = false (A \rightarrow B)

\[ A \rightarrow B \]

R3 = (A, B)
F3 = \{A \rightarrow B\}
Candidate keys = \{A\}
BCNF = true

R4 = (A, C, E)
F4 = {} [[ only trivial ]]
Candidate keys = \{ACE\}
BCNF = true

Dependency preservation ??
We can check:
\[ BC \rightarrow D \text{ (R1)}, \ A \rightarrow B \text{ (R3)}, \]
Dependency-preserving decomposition
Example #4

\[ R = (A, B, C, D, E, H) \]
\[ F = \{ A \rightarrow BC, E \rightarrow HA \} \]
Candidate keys = \{DE\}
BCNF = Violated by \{A \rightarrow BC\} etc…

\[ A \rightarrow BC \]
\[ R1 = (A, B, C) \]
\[ F1 = \{ A \rightarrow BC \} \]
Candidate keys = \{A\}
BCNF = true

\[ E \rightarrow HA \]
\[ R3 = (E, H, A) \]
\[ F3 = \{ E \rightarrow HA \} \]
Candidate keys = \{E\}
BCNF = true

\[ E \rightarrow HA \]
\[ R4 = (ED) \]
\[ F4 = \{ \} \] [[ only trivial ]]\]
Candidate keys = \{DE\}
BCNF = true

Dependency preservation ???
We can check:
\[ A \rightarrow BC \text{ (R1), } E \rightarrow HA \text{ (R3),} \]
Dependency-preserving decomposition
More Examples
Example #5: BCNF Decomposition

- Relation: $R = CSJDPQV$
- FDs: $C \rightarrow CSJDPQV$, $SD \rightarrow P$, $JP \rightarrow C, J \rightarrow S$
- $JP \rightarrow C$ is OK, since JP is a superkey
- $SD \rightarrow P$ is a violating FD
- Decompose into $R_1 = CSJDQV$ and $R_2 = SDP$
- $J \rightarrow S$ is still a violation in $R_1$
- Decompose $R_1$: $CJDQV$ and $JS$
- Final set: $CJDQV$, $JS$, $SDP$
- Order matters: what happens if we use $J \rightarrow S$ first?
Exercise 3

R = \{(A, B, C, D)\).
F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}.

Question 1: Identify all candidate keys for R.

Question 2: Identify the best normal form that R satisfies.

Question 3: Decompose R into a set of BCNF relations.

Question 4: Decompose R into a set of 3NF relations.
Exercise 3 Solution

R = (A, B, C, D).
F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}.

Question 1: Identify all candidate keys for R.

\[ B^+ = B \quad (B \rightarrow B) \]
\[ = BC \quad (B \rightarrow C) \]
\[ = BCD \quad (C \rightarrow D) \]
\[ = ABCD \quad (C \rightarrow A) \]
so the candidate key is B.

B is the ONLY candidate key, because nothing determines B:
There is no rule that can produce B, except B \rightarrow B.
R = (A, B, C, D).
F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}.

Question 2: Identify the best normal form that R satisfies.

R is not 3NF, because:

C \rightarrow D causes a violation,
  C \rightarrow D is non-trivial (\{D\} \not\subseteq \{C\}).
  C is not a superkey.
  D is not part of any candidate key.

C \rightarrow A causes a violation
  Similar to above

B \rightarrow C causes no violation

Since R is not 3NF, it is not BCNF either.
Exercise 3 Solution

R = (A, B, C, D).
F = {C → D, C → A, B → C}.

Question 3: Decompose R into a set of BCNF relations

(1) C → D and C → A both cause violations of BCNF.

Take C → D: decompose R to R₁ = {A, B, C}, R₂ = {C, D}.

(2) Now check for violations in R₁ and R₂. (Actually, using F⁺)

R₁ still violates BCNF because of C → A.

Decompose R₁ to R₁₁ = {B, C}, R₁₂ = {C, A}.

Final decomposition: R₂ = {C, D}, R₁₁ = {B, C}, R₁₂ = {C, A}.

No more violations: Done!
Exercise 3 Solution

R = (A, B, C, D).
F = \{C \rightarrow D, C \rightarrow A, B \rightarrow C\}.

Question 4: Decompose R into a set of 3NF relations.

The canonical cover is F_c = \{C \rightarrow DA, B \rightarrow C\}.
For each functional dependency in F_c we create a table:
R_1 = \{C, D, A\}, R_2 = \{B, C\}.

The table R_2 contains the candidate key for R — we done.
Exercise 4

\[ R = (A, B, C, D) \]
\[ F = \{ AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B \} \]

1. Is R in 3NF, why? If it is not, decompose it into 3NF

2. Is R in BCNF, why? If it is not, decompose it into BCNF
Exercise 4 Solution

\[ R = (A, B, C, D) \]
\[ F = \{AB \rightarrow C, AB \rightarrow D, C \rightarrow A, D \rightarrow B\} \]

1. Is \( R \) in 3NF, why? If it is not, decompose it into 3NF
Yes.

Find all the Candidate Keys:

- \( AB, BC, CD, AD \)

Check all FDs in \( F \) for 3NF condition

2. Is \( R \) in BCNF, why? If it is not, decompose it into BCNF
No. Because for \( C \rightarrow A \), \( C \) is not a superkey. Similar for \( D \rightarrow B \)
\[ R1 = \{C, D\}, R2 = \{A, C\}, R3 = \{B, D\} \]
Summary

• Step 1: BCNF is a good form for relation
  • If a relation is in BCNF, it is free of redundancies that can be detected using FDs.

• Step 2: If a relation is not in BCNF, we can try to decompose it into a collection of BCNF relations.

• Step 3: If a lossless-join dependency-preserving decomposition into BCNF is not possible (or unsuitable given typical queries), consider decomposition into 3NF.

• Note: Decompositions should be carried out while keeping performance requirements in mind.