Functional Dependencies!

3. Extraneous Attributes

Consider $F$, and a functional dependency, $\alpha \rightarrow \beta$.

“Extraneous”: Any attributes in $\alpha$ or $\beta$ that can be safely removed?

Without changing the constraints implied by $F$

- $\sigma$ is extraneous in $\alpha$ if:
  1. $\sigma$ is in $\alpha$, and
  2. $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - \sigma) \rightarrow \beta\}$, $F$ logically implies $F'$ or
     1. let $\alpha = \sigma \gamma$
     2. show $\gamma^*$ includes $\beta$ under $F$

- $A$ is extraneous in $\beta$ if:
  1. $\sigma$ is in $\beta$, and
  2. $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - \sigma)\}$
  3. $F'$ logically implies $F$, or
     1. show $\alpha^*$ includes $\sigma$ under $F'$
3. Extraneous Attributes

Example: Given \( F = \{ A \rightarrow C, AB \rightarrow CD \} \), show \( C \) extra in \( AB \rightarrow CD \)

- \( F' = \{ A \rightarrow C, AB \rightarrow D \} \)
- Need to show \( F' \rightarrow F \), means showing \( AB \rightarrow CD \) given \( F' \)

Using Armstrongs:
- We know:
  - \( AB \rightarrow D \) \( (F') \)
  - \( ABC \rightarrow CD \) \( (aug) \)
- but:
  - \( A \rightarrow C \) \( (F') \)
  - \( AB \rightarrow BC \) \( (aug) \)
  - \( AB \rightarrow ABC \) \( (aug) \)
  - \( AB \rightarrow ABC \rightarrow CD \) \( (trans) \) done.

or attribute closures, show \( \alpha + \) includes \( C \) under \( F' \)
- \( \beta = AB \)
- \( = ABC \) \( (A \rightarrow C) \) done.

3. Extraneous Attributes

Example: Given \( F = \{ A \rightarrow BE, B \rightarrow C, C \rightarrow D, AC \rightarrow DE \} \), remove extraneous attributes

- For left side of \( AC \rightarrow DE \)
  - \( A \) extraneous? \( C^* = CD \), NOT include \( \beta \)
  - \( C \) extraneous? \( A^* = ABCDE \), YES includes \( \beta \)
  - \( F = A \rightarrow BE, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)

- For right side,
  - \( B \) extraneous in \( A \rightarrow BE? \)
    - \( F' = A \rightarrow E, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)
    - no, \( A^+ \) is ADE and not \( B \), so no.
  - \( E \) extraneous in \( A \rightarrow BE? \)
    - \( F' = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)
    - yes, \( A^+ \) is ABCDE includes \( E \), so yes.

- \( F = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)
  - \( D \) extraneous in right side of \( A \rightarrow DE? \)
    - \( F' = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow E \)
    - yes, \( A^+ \) includes \( D \)

- \( F = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow E \)
4. Canonical Cover

- A **canonical cover** for $F$ is a set of dependencies $F_c$ such that
  - $F$ logically implies all dependencies in $F_c$, and
  - $F_c$ logically implies all dependencies in $F$, and
  - No functional dependency in $F_c$ contains an extraneous attribute, and
  - Each left side of functional dependency in $F_c$ is unique

- In some (vague) sense, it is a *minimal* version of $F$:
  
  **repeat**
  
  1. use union rule to merge right sides
  2. eliminate extraneous attributes

- until $F_c$ does not change

---

4. Canonical Cover

- $A \rightarrow B$, $A \rightarrow C$, $C \rightarrow D$, $AC \rightarrow BD$

- **Cover:**
  - $A \rightarrow B$, $A \rightarrow C$, $C \rightarrow D$, $AC \rightarrow BD$
  - $A \rightarrow BC, C \rightarrow D, AC \rightarrow BD$ (union)
    - $a$ extra in $ac \rightarrow bd$?
      - no, $c^* = cd$, doesn’t include “bd”
      - $c$ extra in $ac \rightarrow bd$?
        - yes, $a^* = abcd$, includes “bd”
  - $A \rightarrow BC, C \rightarrow D, A \rightarrow BD$
  - $A \rightarrow BCD, C \rightarrow D$ (union)
    - $b$ extra in $a \rightarrow bcd$? ($F' = a \rightarrow cd, c \rightarrow d$)
      - no $a^* = cd$ in $F'$, not include “bd”
      - $c$ extra in $a \rightarrow bcd$? ($F' = a \rightarrow bd, c \rightarrow d$)
        - no $a^* = bd$ in $F'$, not include “cd”
    - $d$ extra in $a \rightarrow bcd$? ($F' = a \rightarrow bc, c \rightarrow d$)
      - yes, $a^* = bcd$ in $F'$, includes “d”

- $A \rightarrow BC, C \rightarrow D$
Outline

- Mechanisms and definitions to work with FDs
  - Closures, candidate keys, canonical covers etc...
  - Armstrong axioms
- Decompositions
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF
  - How to achieve a BCNF schema
- BCNF may not preserve dependencies
- 3NF: Solves the above problem
- BCNF allows for redundancy
- 4NF: Solves the above problem

Loss-less Decompositions

Definition: A decomposition of \( R \) into \((R_1, R_2)\) is called *lossless* if, for all legal instance of \( r(R) \):

\[
r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \quad \text{or} \quad (\text{select all } R_1 \text{ join } \text{select all } R_2)
\]

or

\[
(\text{select * from (select } R_1 \text{ from } r) \text{ natural join } \text{select } R_2 \text{ from } r)
\]

In other words, projecting on \( R_1 \) and \( R_2 \), and joining back, results in the relation you started with.

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 \( F^+ \).

---

*... R_1 \cap R_2 must be key for R_1 or R_2*
Dependency-preserving Decompositions

- Is it easy to check if dependencies in $F$ hold?
  - Yes if dependencies can be checked in the same table.
- Consider $R = (A, B, C)$, and $F = \{A \rightarrow B, B \rightarrow C\}$

1. Decompose into $R_1 = (A, B)$, and $R_2 = (A, C)$
   - Lossless? Yes.
   - But harder to check for $B \rightarrow C$ as the data is in multiple tables.
2. On the other hand, $R_1 = (A, B)$, and $R_2 = (B, C)$,
   - is both lossless and dependency-preserving

**Definition:**

- Consider decomposition of $R$ into $R_1, \ldots, R_n$.
- Let $F_i$ be the set of dependencies $F^+$ that include only attributes in $R_i$.

- The decomposition is dependency preserving, if
  \[(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+\]
Mechanisms and definitions to work with FDs
  ◦ Closures, candidate keys, canonical covers etc...
  ◦ Armstrong axioms

Decompositions
  ◦ Loss-less decompositions, Dependency-preserving decompositions

BCNF
  ◦ How to achieve a BCNF schema

BCNF may not preserve dependencies

3NF: Solves the above problem

BCNF allows for redundancy

4NF: Solves the above problem

Normalization
Recall that given a relation schema $R$, and a set of functional dependencies $F$, if every FD, $\alpha \rightarrow \beta$, is either:

1. Trivial
2. $\alpha$ is a superkey of $R$

Then, $R$ is in BCNF (Boyce-Codd Normal Form) No redundancy

What if the schema is not in BCNF?

- Decompose (split) the schema into two pieces.
- Careful: you want the decomposition to be lossless

Achieving BCNF Schemas

- For all dependencies $\alpha \rightarrow \beta$ 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 $\alpha \rightarrow \beta$
  - Create $R_1 = \alpha \beta$
  - Create $R_2 = \alpha(R - \beta - \alpha)$
  - Note that: $R_1 \cap R_2 = \alpha$ and $\alpha \rightarrow \alpha \beta (\equiv R_1)$, so this is lossless decomposition

- Repeat for $R_1$, and $R_2$
  - By defining $F_1$ to be all dependencies in $F$ that contain only attributes in $R_1$
  - Similarly $F_2$
Achieving BCNF Schemas

Example 1

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

\[ B \rightarrow C \]

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

R2 = (A, B)
F2 = {A → B}
Candidate keys = {A}
BCNF = true

Dependency preservation???
Yes

Example 2

R = (A, B, C, D, E)
F = {A → B, BC → D}
Candidate keys = {ACE}
BCNF = Violated by {A → B, BC → D}

\[ A \rightarrow B \]

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

R2 = (A, C, D, E)
F2 = {}
Candidate keys = {ACDE}
BCNF = true

Dependency preservation???
No: lost B→CD
Example 3

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

\[ \text{BC} \rightarrow \text{D} \]

\[ \text{R1} = (\text{BCD}) \]
\[ \text{F1} = \{\text{BC} \rightarrow \text{D}\} \]
\[ \text{Candidate keys} = \{\text{BC}\} \]
\[ \text{BCNF} = \text{true} \]

\[ \text{A} \rightarrow \text{B} \]

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

\[ \text{Dependency preservation ???} \]
\[ \text{yes} \]

Example 4

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

\[ \text{A} \rightarrow \text{BC} \]

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

\[ \text{E} \rightarrow \text{HA} \]

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

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

\[ \text{Dependency preservation ???} \]
\[ \text{We can check:} \]
\[ \text{A} \rightarrow \text{BC (R1)}, \text{E} \rightarrow \text{HA (R3)}, \text{Dependency-preserving decomposition} \]
Outline

- Mechanisms and definitions to work with FDs
  - Closures, candidate keys, canonical covers etc...
  - Armstrong axioms
- Decompositions
  - Loss-less decompositions, Dependency-preserving decompositions
- BCNF
  - How to achieve a BCNF schema
- BCNF may not preserve dependencies
- 3NF: Solves the above problem
- BCNF allows for redundancy
- 4NF: Solves the above problem

BCNF may not preserve dependencies

- \( R = \{ J, K, L \} \)
- \( F = \{ JK \rightarrow L, L \rightarrow K \} \)

- Two candidate keys = \( JK \) and \( JL \)

- \( R \) is not in BCNF

- Any decomposition of \( R \) will fail to preserve \( JK \rightarrow L \)

- This implies that testing for \( JK \rightarrow L \) requires a join
BCNF may not preserve dependencies

- Not always possible to find a dependency-preserving decomposition that is in BCNF.

- PTIME to determine if there exists a dependency-preserving decomposition in BCNF
  - in size of $F$

- NP-Hard to find one if it exists

- Better results exist if $F$ satisfies certain properties