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:

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

$\sigma$ is extraneous in $\beta$ if:

1. $\sigma$ is in $\beta$, and
2. $F'$ logically implies $F$,
   - $F' = (F - (\alpha \rightarrow \beta)) \cup (\alpha \rightarrow (\beta - \sigma))$
   - show $\alpha^*$ includes $\sigma$ under $F'$

(\sigma is extraneous in $\alpha$ iff:
- $F \rightarrow F'$, or
- $(\alpha - \sigma)^*$ includes $\beta$ under $F$

$\sigma$ is extraneous in $\beta$ iff:
- $F' \rightarrow F$, or
- $\alpha^*$ includes $\sigma$ in $F'$

show $\alpha^*$ includes $\sigma$ under $F'$
3. Extraneous Attributes

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

\( F' = \{A \rightarrow C, AB \rightarrow D\} \)

» Using Armstrong’s:

\( \text{show } F' \rightarrow F \)

- We know:
  - \( AB \rightarrow D \) (\( F' \))
  - \( ABC \rightarrow CD \) (aug)
- also:
  - \( A \rightarrow C \) (\( F' \))
  - \( AB \rightarrow BC \) (aug w/ \( B \))
  - \( AB \rightarrow ABC \) (aug w/ \( A \))
- then:
  - \( AB \rightarrow ABC \rightarrow CD \) (trans)
  - done.

» Attribute closures (show \( \alpha^+ \) includes \( C \) under \( F' \)):

- \( (AB)^+ = AB \)
- \( = ABC \) (\( A \rightarrow C \))
  - done.

\( \sigma \) is extraneous in \( \alpha \) iff:

\( F \rightarrow F' \), or

\( (\alpha - \sigma)^+ \) includes \( \beta \) under \( F \)

\( \sigma \) is extraneous in \( \beta \) iff:

\( F' \rightarrow F \), or

\( \alpha^+ \) includes \( \sigma \) in \( F' \)

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?
  - NO: \( C^+ = CD \), NOT include \( DE \)
- C extraneous?
  - YES: \( A^+ = ABCDE \), includes \( DE \)

\( F = A \rightarrow BE, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)

» For right side,

- B extraneous in \( A \rightarrow BE \)?
  - \( F' = A \rightarrow E, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)
  - NO: \( A^+ = ADE \), not include \( B \).
- E extraneous in \( A \rightarrow BE \)?
  - \( F' = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)
  - YES: \( A^+ = ABCDE \), includes \( E \).

\( F = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow DE \)

» D extraneous in right side of \( A \rightarrow DE \)?

- \( F' = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow E \)
  - YES: \( A^+ = ABCDE \), so does include \( D \)

\( F = A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow E \)
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 "b"
  - c extra in $a \rightarrow bcd$? ($F' = a \rightarrow bd, c \rightarrow d$)
    - NO: $a^* = bd$ in $F'$, not include "c"
  - d extra in $a \rightarrow bcd$? ($F' = a \rightarrow bc, c \rightarrow d$)
    - YES: $a^* = bcd$ in $F'$, includes "d"
- $A \rightarrow BC, C \rightarrow D$

repeat

1. use union rule to merge right sides
2. eliminate extraneous attributes

until $F_c$ does not change

---

σ is extraneous in α iff:
- $F' \rightarrow F$, or
- $(α - σ)^*$ includes β under F

σ is extraneous in β iff:
- $F' \rightarrow F$, or
- $α^*$ includes σ in $F'$
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 $r(R)$:

$$r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r)$$

or

$$\text{(select all } R_1 \text{ join (select all } R_2))$$

or

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

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 $F$.

or... $R_1 \cap R_2$ 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: $AB \cap AC = A$, which is a key for $R_1$
     - 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

**Dependency-preserving Decompositions**

Definition:
- Consider decomposition of $R$ into $R_1, \ldots, R_n$.
- Let $F_i$ be dependencies using just attributes in $R_i$.

The decomposition is dependency preserving, if

$$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$$
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

Normalization
BCNF

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 $R_2 = R - \beta$
» Note that: $R_1 \cap R_2 = \alpha$ and $\alpha \rightarrow \alpha\beta (R_1)$, 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

### 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 ??
Yes

Dependency preservation ??
No: lost B→CD
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\}

\[ R_1 = (BCD) \]
\[ F_1 = \{ BC \rightarrow D \} \]
Candidate keys = \{BC\}
BCNF = true

\[ R_2 = (A, B, C, E) \]
\[ F_2 = \{ A \rightarrow B \} \]
Candidate keys = \{ACE\}
BCNF = false (A \rightarrow B)

\[ A \rightarrow B \]

\[ R_3 = (A, B) \]
\[ F_3 = \{ A \rightarrow B \} \]
Candidate keys = \{A\}
BCNF = true

\[ R_4 = (A, C, E) \]
\[ F_4 = \{ \} \]  \[\text{[only trivial]}\]
Candidate keys = \{ACE\}
BCNF = true

Dependency preservation ???
yes

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\} and \{E \rightarrow HA\}

\[ A \rightarrow BC \]

\[ R_1 = (A, B, C) \]
\[ F_1 = \{ A \rightarrow BC \} \]
Candidate keys = \{A\}
BCNF = true

\[ R_2 = (A, D, E, H) \]
\[ F_2 = \{ E \rightarrow HA \} \]
Candidate keys = \{DE\}
BCNF = false (E \rightarrow HA)

\[ E \rightarrow HA \]

\[ R_3 = (E, H, A) \]
\[ F_3 = \{ E \rightarrow HA \} \]
Candidate keys = \{E\}
BCNF = true

\[ R_4 = (ED) \]
\[ F_4 = \{ \} \]  \[\text{[only trivial]}\]
Candidate keys = \{DE\}
BCNF = true

Dependency preservation ???
We can check:
A \rightarrow BC (R1), E \rightarrow HA (R3),
Dependency-preserving decomposition
Exam

- Definitions / short answer
- write SQL equations (based on elections, assign 2)
- create E/R diagram, upload picture
  - reduce to relation schema
- relational algebra
  - reading, writing, translating to or from SQL

90 minutes, open book/computer, do your own work. You probably want to have the assignment2 VM up and ready to run some SQL queries.