CMSC424: Database Design
Introduction/Overview

Professor: Pete Keleher
keleher@cs.umd.edu

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
1. Closure

- Given a set of functional dependencies, \( F \), its closure, \( F^+ \), is all FDs that are implied by FDs in \( F \).
  - e.g. If \( A \rightarrow B \), and \( B \rightarrow C \), then clearly \( A \rightarrow C \)

- We can find \( F^+ \) by applying Armstrong’s Axioms:
  - if \( \beta \subseteq \alpha \), then \( \alpha \rightarrow \beta \) (reflexivity)
  - if \( \alpha \rightarrow \beta \), then \( \gamma \alpha \rightarrow \gamma \beta \) (augmentation)
  - if \( \alpha \rightarrow \beta \), and \( \beta \rightarrow \gamma \), then \( \alpha \rightarrow \gamma \) (transitivity)

- These rules are
  - sound (generate only functional dependencies that actually hold)
  - complete (generate all functional dependencies that hold)

Additional rules

- If \( \alpha \rightarrow \beta \gamma \), then \( \alpha \rightarrow \beta \) and \( \alpha \rightarrow \gamma \) (decomposition)
- If \( \alpha \rightarrow \beta \) and \( \alpha \rightarrow \gamma \), then \( \alpha \rightarrow \beta \gamma \) (union)
- If \( \alpha \rightarrow \beta \) and \( \gamma \beta \rightarrow \delta \), then \( \alpha \gamma \rightarrow \delta \) (pseudotransitivity)

- The above rules can be inferred from Armstrong’s axioms.
Example

- \( R = (A, B, C, G, H, I) \)
- \( F = \{ \)
  - \( A \rightarrow B \)
  - \( A \rightarrow C \)
  - \( CG \rightarrow H \)
  - \( CG \rightarrow I \)
  - \( B \rightarrow H \) \}
- Some members of \( F^+ \):
  - \( A \rightarrow H \)
    - by transitivity from \( A \rightarrow B \) and \( B \rightarrow H \)
  - \( AG \rightarrow I \)
    - by augmenting \( A \rightarrow C \) with \( G \), to get \( AG \rightarrow CG \) and then transitivity with \( CG \rightarrow I \)
  - \( CG \rightarrow HI \)
    - by augmenting \( CG \rightarrow I \) to infer \( CG \rightarrow CGI \),
    - and augmenting of \( CG \rightarrow H \) to infer \( CGI \rightarrow HI \),
    - and then transitivity
    - or union

2. Closure of an attribute set

- Given a set of attributes \( \alpha \) and a set of FDs \( F \),
- closure of \( \alpha \) under \( F \) is the set of all attributes implied by \( \alpha \)

- In other words, the largest \( \beta \) such that: \( \alpha \rightarrow \beta \)

- Redefining super keys:
  - The closure of a super key is the entire relation schema

- Redefining candidate keys:
  1. It is a super key
  2. No subset of it is a super key
Computing the closure for $\alpha$

- Simple algorithm

  1. Start with $\beta = \alpha$.
  2. Go over all functional dependencies, $\delta \rightarrow \gamma$, in $F^+$
  3. If $\delta \subseteq \beta$, then
     Add $\gamma$ to $\beta$
  4. Repeat till $\beta$ stops changing

Example

- $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- $(AG)^+ = ?$
  - 1. result = AG
  - 2. result = ABCG (A $\rightarrow$ C and A $\rightarrow$ B)
  - 3. result = ABCGH (CG $\rightarrow$ H and CG $\subseteq$ AGBC)
  - 4. result = ABCGHI (CG $\rightarrow$ I and CG $\subseteq$ AGBCH)

- Is $(AG)$ a candidate key?
  - 1. It is a super key.
  - 2. $(A^+) = ABCH, (G^+) = G.$
  - $YES.$
Uses of attribute set closures

- Determining *superkeys and candidate keys*

- Determining if $\alpha \rightarrow \beta$ is a valid FD
  - Check if $\alpha^+$ contains $\beta$

- Can be used to compute $F^+$

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*

- A is *extraneous* in $\alpha$ if:
  1. $A$ is in $\alpha$, and
  2. $F$ logically implies $(F - \{\alpha \rightarrow \beta\}) \cup ((\alpha - A) \rightarrow \beta)$
  3. $F = \{AB \rightarrow C, A \rightarrow C\}$ clearly $B$ extraneous in $AB \rightarrow C$

  Check if $(\alpha - \beta)^+$ includes $\beta$ under $F$

- A is *extraneous* in $\beta$ if:
  1. $A$ is in $\beta$, and
  2. $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies $F$
  3. $F = \{AB \rightarrow CD, A \rightarrow C\}$ clearly $C$ extraneous in $AB \rightarrow CD$

  Check if $\alpha^+$ includes $A$ under $F'$
3. Extraneous Attributes

A is extraneous in $\alpha$ if:
1. $A$ is in $\alpha$, and
2. $F$ logically implies $(F - (\alpha \rightarrow \beta)) \cup ((\alpha - A) \rightarrow \beta)$
Check if $(\alpha - A)^+ \subseteq \beta$ under $F$

A is extraneous in $\beta$ if:
1. $A$ is in $\beta$, and
2. $F' = (F - (\alpha \rightarrow \beta)) \cup ((\alpha \rightarrow (\beta - A))$ logically implies $F$
Check if $\alpha'$ includes $A$ under $F'$

Example: Given $F = \{A \rightarrow C, AB \rightarrow CD\}$
- Is $C$ extraneous in $AB \rightarrow CD$?
  - $F' = \{A \rightarrow C, AB \rightarrow D\}$
  - $(AB)^+ = ABC$, so yes.

4. Canonical Cover

A canonical cover for $F$ is a set of dependencies $F_c$ such that
1. $F$ logically implies all dependencies in $F_c$, and
2. $F_c$ logically implies all dependencies in $F$, and
3. No functional dependency in $F_c$ contains an extraneous attribute, and
4. 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

\[(A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C)\]
\[(A \rightarrow BC, B \rightarrow C, AB \rightarrow C)\]

1. use union rule to merge right sides
2. eliminate extraneous attributes
until \(F_c\) does not change

**First \(B\) extra?** 
\(F' = (A \rightarrow C, B \rightarrow C, AB \rightarrow C)\)
\((A)^+ = AC\) does not include \(B\)

**\(C\) extra?** 
\(F' = (A \rightarrow B, B \rightarrow C, AB \rightarrow C)\)
\((A)^+ = A\)
\(= AB\)
\(= ABC\) yes, has \(C\)

\(F' = (A \rightarrow B, B \rightarrow C, AB \rightarrow C)\)

**\(A\) extra?** 
\((B)^+ = BC\) yes

\(F' = (A \rightarrow B, B \rightarrow C)\)

\(F' = (A \rightarrow B, B \rightarrow C)\)

**Repeat**

- use union rule to merge right sides
- eliminate extraneous attributes
until \(F_c\) does not change

---

4. Canonical Cover

\(F = \{A \rightarrow BC, B \rightarrow AC, and C \rightarrow AB\}\)

**\(B\) extra?** 
Show \((A)^+\) includes \(B\) under \([A \rightarrow C, B \rightarrow AC, and C \rightarrow AB]\)
\(= A\)
\(= AC\)
\(= ACB\) yes

\([A \rightarrow C, B \rightarrow AC, and C \rightarrow AB]\)

**\(C\) extra?** 
Show \((B)^+\) includes \(C\) under \([A \rightarrow C, B \rightarrow A, and C \rightarrow AB]\)
\(= B\)
\(= AB\)
\(= ABC\) yes

\([A \rightarrow C, B \rightarrow A, and C \rightarrow AB]\)

**\(A\) extra?** 
Show \((C)^+\) includes \(A\) under \([A \rightarrow C, B \rightarrow A, and C \rightarrow B]\)
\(= C\)
\(= BC\)
\(= ABC\) yes

\([A \rightarrow C, B \rightarrow A, and C \rightarrow B]\)

**But not unique!**
\([A \rightarrow C, B \rightarrow C, and C \rightarrow AB]\)
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 \textit{lossless} if, for all legal instance of $r(R)$:
  \[ r = \prod_{R_1}(r) \Join \prod_{R_2}(r) \]
  (select * from (select $R_1$ from $r$) natural join (select $R_2$ 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 \textit{lossless}, iff:
  \[ R_1 \cap R_2 \rightarrow R_1 \quad \text{or} \quad R_1 \cap R_2 \rightarrow R_2 \]
  in $F^+$. \hspace{1cm} \text{or: } R_1 \cap R_2 \text{ must be key for } R_1 \text{ or } R_2
 Dependency-preserving Decompositions

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

1. Decompose into $R_1 = (A, B)$, and $R_2 = (A, C)$
   Lossless? Yes. $R_1 \cap R_2 = \{A\} \rightarrow R_1$
   But, makes it hard to check for $B \rightarrow C$
   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
   Really? What about $A \rightarrow C$?
   If we can check $A \rightarrow B$, and $B \rightarrow C$, $A \rightarrow C$ is implied.

\[ R = \{A, B, C, D, E\} \]
\[ F = \{A \rightarrow B, BC \rightarrow D\} \]
Candidate keys = \{ACE\}  
BCNF = true

\[ R = \{A, B, C, D\} \]
\[ F = \{A \rightarrow B\} \]
Candidate keys = \{A\}  
BCNF = true

\[ R = \{A, C, D\} \]
\[ F = \{AC \rightarrow D\} \]
Candidate keys = \{ACE\}  
BCNF = false ($AC \rightarrow D$)

\[ C \rightarrow D \]

\[ R = \{A, C, E\} \]
\[ F = \{\} \]
Candidate keys = \{ACE\}  
BCNF = true

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

\[ R = \{A, B\} \]
\[ F = \{A \rightarrow B\} \]
Candidate keys = \{A\}  
BCNF = true

\[ R = \{A, C, D\} \]
\[ F = \{AC \rightarrow D\} \]
Candidate keys = \{AC\}  
BCNF = true
Dependency-preserving Decompositions

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

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

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.

- \( \text{PTIME} \) to determine if there exists a dependency-preserving decomposition in BCNF
  - in size of \( F \)

- \( \text{NP-Hard} \) to find one if it exists

- Better results exist if \( F \) satisfies certain properties
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

3NF

- Definition: Prime attributes
  - An attribute that is contained in a candidate key for R

- Example 1:
  - \( R = (A, B, C, D, E, H), F = \{A \rightarrow BC, E \rightarrow HA\}, \)
  - Candidate keys = \{ED\}
  - Prime attributes: D, E

- Example 2:
  - \( R = (J, K, L), F = \{JK \rightarrow L, L \rightarrow K\}, \)
  - Candidate keys = \{JL, JK\}
  - Prime attributes: J, K, L

- Observation/Intuition:
  1. A key has no redundancy (is not repeated in a relation)
  2. A prime attribute has limited redundancy
3NF

$R$ is in *3NF (3\textsuperscript{rd} Normal Form)* if:

- Given a relation schema $R$, and a set of functional dependencies $F$, if every FD, $\alpha \rightarrow \beta$, is either:
  1. Trivial, or
  2. $\alpha$ is a superkey of $R$, or
  3. All attributes in $(\beta - \alpha)$ are prime

- *Why is 3NF good?*
  - Preserves dependencies.

3NF and Redundancy

- *Why does redundancy arise?*
  - Given a FD, $\alpha \rightarrow \beta$, if $\alpha$ is repeated $(\beta - \alpha)$ has to be repeated
    1. If rule 1 is satisfied, $(\beta - \alpha)$ is empty, so not a problem.
    2. If rule 2 is satisfied, then $\alpha$ can't be repeated, so this doesn't happen either
    3. If not, rule 3 says $(\beta - \alpha)$ must contain only prime attributes
      - This limits the redundancy somewhat.

- 3NF relaxes BCNF by allowing some (hopefully limited) redundancy
- *Why good?*
  - There always exists a dependency-preserving lossless decomposition in 3NF.
Decomposing into 3NF

let $F_i$ be a canonical cover for $F$;

$i := 0$;

for each functional dependency $\alpha \rightarrow \beta$ in $F_i$

$i := i + 1$;

$R_i := \alpha \beta$;

if none of the schemas $R_j$, $j = 1, 2, \ldots, i$ contains a candidate key for $R$

then

$i := i + 1$;

$R_i :=$ any candidate key for $R$;

/* Optionally, remove redundant relations */

repeat

if any schema $R_j$ is contained in another schema $R_k$

then

/* Delete $R_j$ */

$R_j := R_k$;

$i := i - 1$;

until no more $R_j$s can be deleted

return ($R_1, R_2, \ldots, R_i$)

Figure 8.12  Dependency-preserving, lossless decomposition into 3NF.

3CNF Example

$\bullet$ $(R) = (A,B,C,D,E,F,G,H)$

$\bullet$ Function Dependencies

$\bullet$ $F = \{A \rightarrow CGH, AD \rightarrow C, DE \rightarrow F, G \rightarrow H\}$

$\bullet$ $R_1 = \{ACG\}, R_2 = \{ADC\}, R_3 = \{DEF\}, R_4 = \{GH\}$

$\bullet$ $R_1 = \{ACG\}, R_2 = \{ADC\}, R_3 = \{DEF\}, R_4 = \{GH\}, R_5 = \{ABDE\}$

$\bullet$ $R_1 = \{ACG\}, R_2 = \{ADC\}, R_3 = \{DEF\}$, $R_5 = \{ABDE\}$

$\bullet$ $F' =$

$\bullet$ $\{A \rightarrow CGH, AD \rightarrow C, DE \rightarrow F, G \rightarrow H\}$

$\bullet$ $\{A \rightarrow CGH, AD \rightarrow C, DE \rightarrow F, G \rightarrow H\}$

$\bullet$ $\{A \rightarrow CG, DE \rightarrow F, G \rightarrow H\}$

$\bullet$ $H$ is extra in $A \rightarrow CGH$, D extra in $AD \rightarrow C$ - then merge w/ $A \rightarrow CG$

$\bullet$ $R_1 = \{ACG\}, R_3 = \{DEF\}, R_4 = \{GH\}$

$\bullet$ $R_1 = \{ACG\}, R_2 = \{DEF\}, R_3 = \{GH\}, R_5 = \{ABDE\}$

$\bullet$ Lossless: Each has a single FD that is a key

$\bullet$ Preserves dependencies: each carried through a single subrelation