Closure and Lossless Decomposition
Boyce-Codd Normal Form

- A relation schema $R$ is in BCNF if for all functional dependencies in $F^+$ of the form $\alpha \rightarrow \beta$ at least one of the following holds
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - $\alpha$ is a superkey for $R$

- $bor\_loan = (customer\_id, loan\_number, amount)$ is not in BCNF
  - $loan\_number \rightarrow amount$ holds on $bor\_loan$ but $loan\_number$ is not a superkey
Decomposing into BCNF

• For schema R and a non-trivial dependency \( \alpha \rightarrow \beta \) causing a violation of BCNF, decompose R into
  
  – \((\alpha \cup \beta)\): \(\alpha\) is the key
  – \((R - (\beta - \alpha))\)

• \textit{bor_loan} = (\textit{customer_id, loan_number, amount}), \textit{loan_number} \rightarrow \textit{amount}
  
  – (loan_number, amount)
  – (customer_id, loan_number)
Third Normal Form

• A relation schema $R$ is in the third normal form (3NF) if for all $\alpha \rightarrow \beta$ in $F^+$ at least one of the following holds
  – $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \in \alpha$)
  – $\alpha$ is a superkey for $R$
  – Each attribute $A$ in $\beta - \alpha$ is contained in a candidate key for $R$
    • Each attribute may be in a different candidate key
• If a relation is in BCNF it is in 3NF
  – In BCNF one of the first two conditions above must hold
• The third condition is a minimal relaxation of BCNF to ensure dependency preservation
Comparison of BCNF and 3NF

• It is always possible to decompose a relation into a set of relations that are in 3NF such that the decomposition is lossless and the dependencies are preserved.

• It is always possible to decompose a relation into a set of relations that are in BCNF such that the decomposition is lossless.
  – It may not be possible to preserve all functional dependencies.
Using BCNF and 3NF

• How can we generate lossless decompositions into BCNF and 3NF?
• How can we test if a decomposition is dependency-preserving?
• Some critical operations
  – If $\alpha \rightarrow \beta$, is $\alpha$ a superkey?
  – To preserve all functional dependencies $F$, what is the minimal set of functional dependencies that we need to preserve?
Closure of Attribute Sets

- Given a set of attributes \( \alpha \), the closure of \( \alpha \) under \( F \) (denoted by \( \alpha^+ \)) is the set of attributes that are functionally determined by \( \alpha \) under \( F \)
  - Application: whether \( \alpha \) is a super key?
- Algorithm to compute \( \alpha^+ \), the closure of \( \alpha \) under \( F \)

\[
\text{result} := \alpha; \\
\text{while} \ (\text{changes to result}) \ \text{do} \\
\quad \text{for each} \ \beta \rightarrow \gamma \ \text{in} \ F \ \text{do} \\
\quad \quad \text{begin} \\
\quad \quad \quad \text{if} \ \beta \subseteq \text{result} \ \text{then} \ \text{result} := \text{result} \cup \gamma \\
\quad \quad \text{end}
\]
Example of Attribute Set Closure

- $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$
- $(AG)^+$
  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?
  - Is AG a super key?
    - Does AG $\rightarrow$ R?
      - Yes, $(AG)^+ \supseteq R$
    - Is any subset of AG a superkey?
      - Does A $\rightarrow$ R? Is $(A)^+ \supseteq R$?
      - Does G $\rightarrow$ R? Is $(G)^+ \supseteq R$?
Uses of Attribute Closure

• Testing for superkey: check if $\alpha^+$ contains all attributes of $R$

• Testing functional dependencies
  – To check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in $F^+$), just check if $\beta \subseteq \alpha^+$

• Computing closure of $F$
  – For each $\gamma \subseteq R$, find the closure $\gamma^+$
  – For each $S \subseteq \gamma^+$, output a functional dependency $\gamma \rightarrow S$
Armstrong’s Axioms

• Finding F⁺
  – (reflexivity) If $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
  – (augmentation) If $\alpha \rightarrow \beta$, then $\gamma \alpha \rightarrow \gamma \beta$
  – (transitivity) If $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$

• These rules are
  – Sound: generate only functional dependencies that actually hold
  – Complete: generate all functional dependencies that hold
Procedure for Computing $F^+$

$F^+ = F$
repeat
  for each functional dependency $f$ in $F^+$
    apply reflexivity and augmentation rules on $f$
    add the resulting functional dependencies to $F^+$
  for each pair of functional dependencies $f_1$ and $f_2$ in $F^+$
    if $f_1$ and $f_2$ can be combined using transitivity
      then add the resulting functional dependency to $F^+$
  until $F^+$ does not change any further
Redundancy among Dependencies

• \( A \rightarrow C \) is redundant in: \( \{A \rightarrow B, \ B \rightarrow C\} \)
• Parts of a functional dependency may be redundant
  – On left hand side of a rule: \( \{A \rightarrow B, \ B \rightarrow C, \ AC \rightarrow D\} \)
    can be simplified to \( \{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\} \)
  – On right hand side of a rule: \( \{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow CD\} \)
    can be simplified to \( \{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\} \)

• A **canonical cover** of \( F \) is a “minimal” set of functional dependencies equivalent to \( F \), having no redundant dependencies or redundant parts of dependencies
Extraneous Attributes

• Consider functional dependency $\alpha \rightarrow \beta$ in $F$
  – Attribute A is extraneous in $\alpha$ if $A \in \alpha$ and $F$ logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  – Attribute A is extraneous in $\beta$ if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies $F$
• Implication in the opposite direction is trivial in each of the cases above, since a “stronger” functional dependency always implies a “weaker” one
Extraneous Attributes – Example

• Example: Given $F = \{ A \rightarrow C, AB \rightarrow C \}$
  
  $B$ is extraneous in $AB \rightarrow C$ because $\{ A \rightarrow C, AB \rightarrow C \}$ logically implies $A \rightarrow C$ (i.e. the result of dropping $B$ from $AB \rightarrow C$)

• Example: Given $F = \{ A \rightarrow C, AB \rightarrow CD \}$
  
  $C$ is extraneous in $AB \rightarrow CD$ since $AB \rightarrow C$ can be inferred even after deleting $C$
Testing Extraneous Attributes

- Consider functional dependency $\alpha \rightarrow \beta$ in $F$
- To test if attribute $A \in \alpha$ is extraneous in $\alpha$
  - Compute $(\alpha - A)^+$ using the dependencies in $F$
  - If $(\alpha - A)^+$ contains $A$, $A$ is extraneous
- To test if attribute $A \in \beta$ is extraneous in $\beta$
  - Compute $\alpha^+$ using only the dependencies in $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$,
  - if $\alpha^+$ contains $A$, $A$ is extraneous
Canonical Cover

- A canonical cover for $F$ is a set of dependencies $F_c$ such that
  - $F$ and $F_c$ are logically equivalent to each other
    - $F$ logically implies all dependencies in $F_c$
    - $F_c$ logically implies all dependencies in $F$
  - No functional dependency in $F_c$ contains an extraneous attribute, and
  - Each left side of functional dependency in $F_c$ is unique
Canonical Cover Computation

repeat
  use the union rule to replace any dependencies in F
  $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$
  find a functional dependency $\alpha \rightarrow \beta$ with an extraneous attribute either in $\alpha$ or in $\beta$
  if an extraneous attribute is found,
  then delete it from $\alpha \rightarrow \beta$
  until F does not change
• The union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied
Example

- $R = (A, B, C)$
  - $F = \{A \rightarrow BC, B \rightarrow C, A \rightarrow B, AB \rightarrow C\}$
- Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$
  - Now, $F = \{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- $A$ is extraneous in $AB \rightarrow C$
  - $B \rightarrow C$ is already present
  - Now, $F = \{A \rightarrow BC, B \rightarrow C\}$
  - $C$ is extraneous in $A \rightarrow BC$
    - Check if $A \rightarrow C$ is logically implied by $A \rightarrow B$ and the other dependencies
      - Yes: using transitivity on $A \rightarrow B$ and $B \rightarrow C$.
      - Can use attribute closure of $A$ in more complex cases
- The canonical cover is: $\{A \rightarrow B, B \rightarrow C\}$
Lossless-join Decomposition

• If $R$ is decomposed into $R_1$ and $R_2$, we require that for all possible relations $r$ on schema $R$ satisfies $r = \prod_{R_1}(r) \bowtie \prod_{R_2}(r)$

• A decomposition of $R$ into $R_1$ and $R_2$ is **lossless join** if and only if at least one of the following dependencies is in $F^+$
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$
Example

- \( R = (A, B, C), F = \{A \rightarrow B, B \rightarrow C\} \)
  - Can be decomposed in two different ways
- \( R_1 = (A, B), \; R_2 = (B, C) \)
  - Lossless-join decomposition:
    \[ R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC \]
  - Dependency preserving
- \( R_1 = (A, B), \; R_2 = (A, C) \)
  - Lossless-join decomposition:
    \[ R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB \]
  - Not dependency preserving
    (cannot check \( B \rightarrow C \) without computing \( R_1 \Join R_2 \))
Dependency Preservation

• Let $F_i$ be the set of dependencies $F^+$ that includes only attributes in $R_i$
  – A decomposition is dependency preserving, if $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$
  – If it is not, then checking updates for violation of functional dependencies may require computing joins, which is expensive
Testing Dependency Preservation

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of $R$ into $R_1, R_2, \ldots, R_n$
  
  $result = \alpha$
  repeat
  for each $R_i$ in the decomposition
  
  $t = (result \cap R_i)^+ \cap R_i$, result = result $\cup t$
  
  until result does not change
  
  - If $result$ contains all attributes in $\beta$, then the functional dependency $\alpha \rightarrow \beta$ is preserved

- We apply the test on all dependencies in $F$ to check if a decomposition is dependency preserving

- This procedure takes polynomial time
  
  - Compute $F^+$ and $(F_1 \cup F_2 \cup \ldots \cup F_n)^+$ requires exponential time
Example

- \( R = (A, B, C) \)
  \( F = \{A \rightarrow B, B \rightarrow C\} \)
  Key = \{A\}
- \( R \) is not in BCNF
- Decomposition \( R_1 = (A, B), \ R_2 = (B, C) \)
  - \( R_1 \) and \( R_2 \) are in BCNF
  - Lossless-join decomposition
  - Dependency preserving
Summary and To-Do List

• Closure of attribute sets: concept, computation, and applications
• Canonical cover: concept and computation
• Lossless join decomposition
• Testing dependency preservation
• Read Sections 7.4.2-7.4.5
• Assignment 2