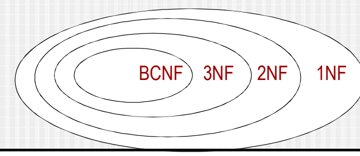
Normal Forms

- If a relation schema is in a normal forms, we know that it is in some particular shape/health in the sense that *certain kinds of problems and issues (related to redundancy) will not arise*
- Given a relation schema **R**, we need to determine if it is in certain normal form. If it is not, we need methods to decompose it into smaller such normal relations. How?
- To address these issues, we study **normal forms**

1

Normal Forms

- The normal forms as defined and captured by FD's:
  - First normal form (1NF)
  - Second normal form (2NF)
  - √ Third normal form (3NF)
  - √ Boyce-Codd normal form (BCNF)
- The relationships among these normal forms:



2

Third Normal Form (3NF)

Let **R** be a relation schema with a set of FD's **F**.

- We say **R** w.r.t. **F** is in 3NF (**third normal form**), if for every FD **X** → **A** in **F**, at least one of the following conditions holds:
  - **X** → **A** is a trivial, i.e., **A** ∈ **X,** or
  - **X** is a superkey, or
  - **X** is not a key but **A** is part of some key of **R**

➔ Therefore, to determine if **R** is in 3NF w.r.t. F, we need to:
  - Check if the LHS of each nontrivial FD in **F** is a superkey
  - If not, check if its RHS is part of any key of **R**

3

Boyce-Codd Normal Form

Given: A relation schema **R** with a set of FD's **F** on **R.**

- We say **R** w.r.t. **F** is in **Boyce-Codd normal form**, if for every FD **X** → **A** in **F**, at least one of the following conditions holds:
  - **A** ∈ **X**, that is, **X** → **A** is a trivial FD, or
  - **X** is a superkey
- To determine if **R** is in BCNF w.r.t. **F**,

  For every FD **X** → **A**, check if its LHS **X** is a superkey.

  For any FD **X** → **A** in **F,** if there is an attribute B of **R** that is not in X^+, then **R** is not in BCNF.

4

Decomposition into BCNF

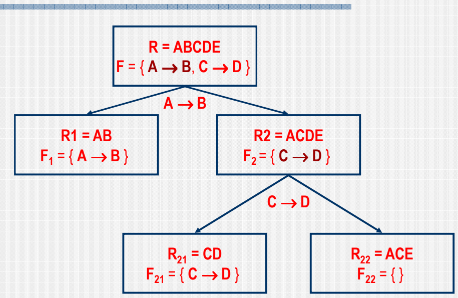
- Consider <**R**, **F**>, where **R** is in 1NF.
- If **R** is not in BCNF, we can always obtain a *lossless-join decomposition* of **R** into a collection of BCNF relations
- However, this decomposition may not always be dependency preserving.
- The basic step of a BCNF algorithm (done recursively):

  Pick every FD **X** → **A** ∈ **F** that violates the BCNF requirement:

  1. Decompose **R** into two relations: **XA** and **R – A**
  2. If either **R–A** or **XA** is not in BCNF, decompose it further

5

Example (Decomposition into BCNF relations)



6

Decomposition into 3NF

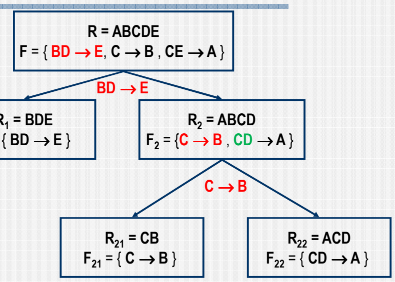
- We can always obtain a lossless-join, dependency-preserving decomposition of a relation into 3NF relations. How?
- We discuss 2 solution approaches for 3NF decomposition.
- **Approach 1:** using the *binary decomposition* method.

Let **R** = { R_1, R_2, . . . R_n} be the result. Recall that this is always lossless-join, but may not preserve all the FD's➔ need to fix this!

- Identify the set **N** of FD's in **F** which we lost in the decomposition proc.
- For each FD **X** $\rightarrow$ **A** in **N**, create a relation schema **XA** and add it to **R**
- A refinement step to avoid creating MANY relations: if there are several FD's with the same LHS, e.g., $X \rightarrow A_1$, $X \rightarrow A_2$, . . . , $X \rightarrow A_k$, create **just one relation** with schema $XA_1 \ldots A_k$

7

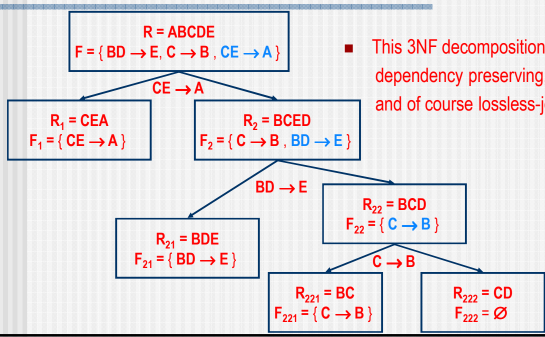
Example (3NF Decomposition)



- **CE** $\rightarrow$ **A** is not preserved, since **A** $\notin$ {**CE**}$^+$ w.r.t. $F_1 \cup F_{21} \cup F_{22}$

➔ To fix this, We add to **R**, a new relation R_3 = **CEA** with F_3 = {**CE** $\rightarrow$ **A** }

Example (using a *different order*)



- This 3NF decomposition is dependency preserving, and of course lossless-join

9

Decomposition into 3NF

- 1ST approach (binary decomposition):
  - Lossless-join √
  - May not be dependency preserving. If so, then add extra relations **XA**, for every FD **X** $\longrightarrow$ **A** we lost
- **Approach 2:** the *synthesis* approcah
  - Dependency preservation √
  - However, may not be lossless-join. If so, we must add to **R**, one extra relation that includes whose attributes form a key of **R**

  What would be the FDs on this newly added relation?

10

Decomposition into 3NF (Using the synthesis approach)

Consider <**R, F**>

- The synthesis approach:
  - Get a minimal cover F^c of **F**
  - For each FD **X** $\rightarrow$ **A** in F^c, add schema **XA** to **R**
  - If the decomposition **R** is not lossless,

    add to **R** an extra relation containing any key of **R**

11

Example

- R = (**A**, **B**, **C**)
- F = { **A** $\rightarrow$ **B**, **C** $\rightarrow$ **B** }
- Decompose R into R_1 = (**A**, **B**) and R_2 = (**B**, **C**)
- This decomposition is not lossless

  ➔ Add R_3 = (**A**, **C**)
- The decomposition **R** = {R_1, R_2, R_3} is both lossless and dependency-preserving

12

An Algorithm to Check Lossless join

Suppose relation $\mathbf{R}(A_1, \ldots, A_k)$ is decomposed into $R_1, \ldots, R_n$
To determine if this decomposition is lossless, we use a table,
$L[1 \ldots n][1 \ldots k]$

Initializing the table:

for each relation R_i **do**
  for each attribute A_j **do**
    if A_j is an attribute in R_i
      then $L[i][j] \leftarrow a_j$
      else $L[i][j] \leftarrow b_{ij}$

13

Algorithm to Check Lossless (cont'd)

repeat
  for each FD $X \to Y$ in **F do:**
    if $\exists$ rows i and j such that $L[i] == L[j]$, for each attribute in **X**,
      then for $\forall$ column t corresponding to an attribute A_t in **Y do:**
        if $L[i][t] == a_t$
          then $L[j][t] \leftarrow a_t$
        else if $L[j][t] == a_t$
          then $L[i][t] \leftarrow a_t$
        else $L[j][t] \leftarrow L[i][t]$
until *no change*

The decomposition is lossless if, after performing this algorithm, L contains a row of all a's. That is, if there exists a row i in L such that: $L[i][j] == a_j$ for every column j corresponding to each attribute A_j in **R**

14

Examples

- Given $\langle \mathbf{R}, \mathbf{F} \rangle$, where $\mathbf{R} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, and $\mathbf{F} = \{ \mathbf{A} \to \mathbf{B}, \mathbf{A} \to \mathbf{C}, \mathbf{C} \to \mathbf{D} \}$ is a set of FD's on **R**
- Is the decomposition $\underline{\mathbf{R}} = \{R_1, R_2\}$ lossless, where $R_1 = (\mathbf{A}, \mathbf{B}, \mathbf{C})$ and $R_2 = (\mathbf{C}, \mathbf{D})$?
  - To be discussed in class
- Now consider $\mathbf{S} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \mathbf{E})$ with the FD's:
  $\mathbf{G} = \{ \mathbf{AB} \to \mathbf{CD}, \ \mathbf{A} \to \mathbf{E}, \mathbf{C} \to \mathbf{D} \}$
- Is decomposition of $\underline{\mathbf{S}} = \{S_1, S_2, S_3\}$ lossless, where $S_1 = (\mathbf{A}, \mathbf{B}, \mathbf{C})$, $S_2 = (\mathbf{B}, \mathbf{C}, \mathbf{D})$, and $S_3 = (\mathbf{C}, \mathbf{D}, \mathbf{E})$?
  - To be discussed in class

15

Checking if a decomposition is Dependency-Preserving?

Inputs: Let $\langle \mathbf{R}, \mathbf{F} \rangle$, where $\mathbf{F} = \{X_1 \to Y_1, \ldots, X_n \to Y_n\}$.
  Suppose $\underline{\mathbf{R}} = \{ \mathbf{R}_1, \ldots, \mathbf{R}_k \}$ is a decomposition of **R**
  and $\mathbf{F}_i$ is the projection of **F** on schema $\mathbf{R}_i$

Method:
  preserved $\leftarrow$ *TRUE*
  for each FD $\mathbf{X} \to \mathbf{Y}$ in **F** and while preserved == *TRUE*
  do compute $\mathbf{X}^+$ under $\mathbf{F}_1 \cup \ldots \cup \mathbf{F}_k$;
    if $\mathbf{Y} \not\subseteq \mathbf{X}^+$ then {*preserved* $\leftarrow$ *FALSE; exit* };
  end

16

Example

- Consider $\mathbf{R} = (\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$, $\mathbf{F} = \{ \mathbf{A} \to \mathbf{B}, \mathbf{B} \to \mathbf{C}, \mathbf{C} \to \mathbf{D} \}$
- Is the decomposition $\underline{\mathbf{R}} = \{R_1, R_2\}$ dependency-preserving, where
  $\mathbf{R}_1 = (\mathbf{A}, \mathbf{B})$, $\mathbf{F}_1 = \{ \mathbf{A} \to \mathbf{B} \}$, $\mathbf{R}_2 = (\mathbf{A}, \mathbf{C}, \mathbf{D})$, AND $\mathbf{F}_2 = \{ \mathbf{C} \to \mathbf{D}, \mathbf{A} \to \mathbf{D}, \mathbf{A} \to \mathbf{C} \}$?
  - Check if $\mathbf{A} \to \mathbf{B}$ is preserved
    - Compute $\mathbf{A}^+$ under $\{ \mathbf{A} \to \mathbf{B} \} \cup \{ \mathbf{C} \to \mathbf{D}, \mathbf{A} \to \mathbf{D}, \mathbf{A} \to \mathbf{C} \}$
      - $\mathbf{A}^+ = \{ \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D} \}$
      - Check if $\mathbf{B} \in \mathbf{A}^+$
      - **Yes**
    - $\mathbf{A} \to \mathbf{B}$ is preserved
  - Check if $\mathbf{B} \to \mathbf{C}$ is preserved
    - Compute $\mathbf{B}^+$ under $\{ \mathbf{A} \to \mathbf{B} \} \cup \{ \mathbf{C} \to \mathbf{D}, \mathbf{A} \to \mathbf{D}, \mathbf{A} \to \mathbf{C} \}$
      - $\mathbf{B}^+ = \{ \mathbf{B} \}$
    - Check if $\mathbf{C} \in \mathbf{B}^+$
      - **No**
    - $\mathbf{B} \to \mathbf{C}$ is not preserved

➔The decomposition is not dependency-preserving

17