Database Management Systems
Course 236363
Lecture 7:
Schema Normalization

Schema Anomalies

- Redundant storage
  - Repeatedly storing the same information
- Update anomaly
  - To change a repeated item, every occurrence should be changed
- Insertion anomaly
  - Some information cannot be stored without additional (possibly unavailable) information
- Deletion anomaly
  - Some information cannot be deleted without deleting additional (possibly desired) information

From ERD to Normalization

- We have learned how to design schemas using ERDs
- But it is often not enough for a proper translation into well designed relations
- ERD is limited in constraint representation; we need a more careful design to enforce such constraints
- It may be challenging to avoid anomalies when dependencies are complicated

Example

The Refined Design Process (Normalization)

1. Define the involved attributes
2. Determine what constraints / dependencies hold in real life
3. Decide on desired properties
4. Decompose into multiple good (“normalized”) schemas

What makes a decomposition “good”? Are there principles to follow? Can design be automated?
Outline

- Introduction
  - Normal Forms
    - BCNF
    - 3NF
- Decomposition
  - NF Decompositions
  - Preserving Data
  - Preserving Dependencies
- Decomposition Algorithms
  - 3NF
  - BCNF (complementary)
  - Note on 4NF (complementary)

Notation

- During this lecture we view a relation schema as a pair \((U,F)\) where:
  - \(U\) is a finite set of attributes
  - \(F\) is a set of FDs over \(U\)
- In particular, we ignore:
  - relation names
  - order among attributes

Basic Terminology

- Let \((U,F)\) be a relation schema
- Recall: A superkey is a set \(K\) of attributes such that \(K^+\) contains every attribute in \(U\)
- Recall: A key is a superkey \(K\) that does not contain any other superkey
  - That is, if \(Y\subseteq K\) then \(Y\) is not a superkey
- Attributes of keys are called prime
- “Schema normalization” constrains the relationship between FDs, keys, prime attributes and nonprime attributes

History of Normal Forms

- 1970: 1NF
  - Nonprime attributes are not dependent on strict parts of key attributes
- 1971: BCNF
  - “Standard” normal form: a nonprime attribute can be determined only by a superkey
- 1974: 3NF
  - A relation does not involve any “implicit” joins
- 1977: 4NF
  - No nontrivial MVDs except for superkeys
- 1979: 5NF
  - No nontrivial FDs except for superkeys

Our Focus

- We mainly focus on BCNF and 3NF
  - Historically BCNF came after 3NF, but we start with BCNF since it is simpler
- Complementary material: 4NF
Boyce-Codd Normal Form (BCNF)

- A schema \((U, F)\) is in **BCNF** if every nontrivial FD implied by \(F\) has a superkey on its premise (lhs)
- That is, every \(X \rightarrow Y\) in \(F^+\) is such that
  - \(X\) is a superkey; or
  - \(Y \subseteq X\)

Can BCNF be Tested Efficiently?

- On the face of it, we need to consider every derived FD (exponentially many); however:
  - **THEOREM**: The following are equivalent:
    1. The schema \((U, F)\) is in BCNF (i.e., every nontrivial \(X \rightarrow Y\) in \(F^+\) is such that \(X\) is a superkey)
    2. In every nontrivial \(X \rightarrow Y\) in \(F\), \(X\) is a superkey
- Hence, it suffices to check only \(F\)
- Proof not given
  - *But which direction is straightforward?*
  - *So what would be an efficient BCNF testing?*

Third Normal Form (3NF)

- Recall: an attribute \(A\) is *prime* if it is a part of some key
  - Note: “key” as opposed to “superkey;” every attribute belongs to some superkey (e.g., all attributes)
- A schema is in **3NF** if for every nonprime \(A\) and nontrivial derived \(X \rightarrow A\), the set \(X\) is a superkey
- Equivalently, for every \(X \rightarrow A\) in \(F^+\) at least one of the following holds:
  - \(X\) is a superkey
  - \(A \in X\)
  - \(A\) is prime

Outline

- Introduction
- Normal Forms
  - BCNF
  - 3NF
- Decomposition
  - NF Decompositions
  - Preserving Data
  - Preserving Dependencies
- Decomposition Algorithms
  - 3NF
  - BCNF (complementary)
  - Note on 4NF (complementary)

Examples

```
<table>
<thead>
<tr>
<th>Keys</th>
<th>Faculty: name, symbol, dean</th>
<th>BCNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>symbol</td>
<td></td>
</tr>
<tr>
<td>symbol</td>
<td>dean</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Social network: follows, followed, fid</th>
<th>BCNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>follows</td>
<td>followed</td>
<td></td>
</tr>
<tr>
<td>followed</td>
<td>fid</td>
<td></td>
</tr>
<tr>
<td>fid</td>
<td>follows, followed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Address: state, city, street, zip</th>
<th>3NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>city, street</td>
<td></td>
</tr>
<tr>
<td>city</td>
<td>street</td>
<td></td>
</tr>
<tr>
<td>street</td>
<td>zip</td>
<td></td>
</tr>
<tr>
<td>zip</td>
<td>state</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Tracks: track, faculty, consultant, campus</th>
<th>3NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>track</td>
<td>faculty, consultant</td>
<td></td>
</tr>
<tr>
<td>faculty</td>
<td>consultant</td>
<td></td>
</tr>
<tr>
<td>consultant</td>
<td>campus</td>
<td></td>
</tr>
<tr>
<td>campus</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Examples

```
<table>
<thead>
<tr>
<th>Keys</th>
<th>Track: track, faculty, consultant, campus</th>
</tr>
</thead>
<tbody>
<tr>
<td>track</td>
<td>faculty, consultant</td>
</tr>
<tr>
<td>faculty</td>
<td>consultant</td>
</tr>
<tr>
<td>consultant</td>
<td>campus</td>
</tr>
<tr>
<td>campus</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Faculty: name, symbol, dean</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>symbol</td>
</tr>
<tr>
<td>symbol</td>
<td>dean</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Social network: follows, followed, fid</th>
</tr>
</thead>
<tbody>
<tr>
<td>follows</td>
<td>followed</td>
</tr>
<tr>
<td>followed</td>
<td>fid</td>
</tr>
<tr>
<td>fid</td>
<td>follows, followed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Address: state, city, street, zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>city, street</td>
</tr>
<tr>
<td>city</td>
<td>street</td>
</tr>
<tr>
<td>street</td>
<td>zip</td>
</tr>
<tr>
<td>zip</td>
<td>state</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Keys</th>
<th>Tracks: track, faculty, consultant, campus</th>
</tr>
</thead>
<tbody>
<tr>
<td>track</td>
<td>faculty, consultant</td>
</tr>
<tr>
<td>faculty</td>
<td>consultant</td>
</tr>
<tr>
<td>consultant</td>
<td>campus</td>
</tr>
<tr>
<td>campus</td>
<td></td>
</tr>
</tbody>
</table>
```
Testing 3NF?

- The following algorithm works:
  - For every nontrivial FD $X \rightarrow Y$ in $F$
    1. Check whether $X$ is a superkey
    2. Check whether every attribute in $Y \setminus X$ is prime
  - As we know, (1) can be tested efficiently
  - What about (2)?
    - It is NP-complete! (unlikely to be solvable in polynomial time)
  - And in fact, testing whether a schema is in 3NF is an NP-complete problem [Jou, Fischer 82]

Outline

- Introduction
- Normal Forms
  - BCNF
  - 3NF
- Decomposition
  - NF Decompositions
  - Preserving Data
  - Preserving Dependencies
- Decomposition Algorithms
  - 3NF
  - BCNF (complementary)
  - Note on 4NF (complementary)

Decomposition

- We can fix a “badly designed” schema by decomposing it into several smaller schemas
- But we need to do so correctly!
  - Do not change our intended information
  - Do not violate the FDs
  - Attain a normal form
- In this part, we will make the above formal
- First, we need a notation

Restricting a Set of FDs

- Let $(U,F)$ be a schema
- Let $W$ be a subset of $U$
- We denote by $F[W]$ the set of all the FDs $X \rightarrow Y$ in $F$ such that $XY \subseteq W$

Formal Definition

- A decomposition of a schema $(U,F)$ is a collection $(X_1,F_1), \ldots, (X_k,F_k)$ of schemas such that:
  - $U = X_1 \cup \cdots \cup X_k$
    - That is, the $X_i$ cover all the attributes in $U$
  - For $i=1,\ldots,k$ we have $(F)^+ = F^+[X_i]$
    - That is, each $F_i$ consists of the FDs imposed by $F$ on $X_i$

Decomposing and Composing Relations
Representing $F_i$

- Given the schema $(U,F)$, it suffices to represent a decomposition using the collection $\{X_1, \ldots, X_k\}$ without mentioning the FDs $F_i$.
- Since $F_i$ can be anything equivalent to $F^+[X_i]$.
- Problem: naively constructing $F_i$ as $F^+[X_i]$ can be expensive, since $F^+$ and $F^+[X_i]$ can be exponentially larger than $U$.
  - This problem is unavoidable: It may be that $F^+[X_i]$ is not equivalent to any sub-exponential #FDs!
  - We keep this problem in mind – we will not assume that $F^+[X_i]$ can be materialized efficiently.

Obtaining Normal Forms

- Let $N$ be a normal form (e.g., 3NF, BCNF).
- An $N$ decomposition of a schema $(U,F)$ is a decomposition $\{X_1, \ldots, X_k\}$ of $(U,F)$ such that each $(X_i, F^+[X_i])$ is in $N$.
- We will discuss 3NF decompositions and BCNF decompositions.

Examples

3NF decomposition? BCNF decomposition?

- ABCD $\rightarrow$ AD
  - $A \rightarrow B$, $B \rightarrow C$, $ABC \rightarrow D$, $D \rightarrow B$
  - Answer: BCNF, 3NF

- ABCD $\rightarrow$ BC
  - $A \rightarrow C$, $B \rightarrow C$
  - Answer: 3NF, not BCNF

- ABCD $\rightarrow$ BD
  - $A \rightarrow B$, $D \rightarrow B$
  - Answer: not 3NF, not BCNF

Good Decomposition?

<table>
<thead>
<tr>
<th>person</th>
<th>building</th>
<th>room</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alma</td>
<td>Taub</td>
<td>152</td>
</tr>
<tr>
<td>Amir</td>
<td>Meyer</td>
<td>35</td>
</tr>
<tr>
<td>Ahuva</td>
<td>Meyer</td>
<td>246</td>
</tr>
</tbody>
</table>

$\pi_{\text{person} \rightarrow \text{building}, \text{room}}$
Good Decomposition?

Can you restore?

Lossless Decomposition

- Let \( \{X_1, \ldots, X_k\} \) be a decomposition of \((U,F)\)
- We say that \( \{X_1, \ldots, X_k\} \) is a lossless decomposition of \((U,F)\) if for all relations \( r \) over \((U,F)\) we have:
  \[
  \pi_{X_1}(r) \Join \cdots \Join \pi_{X_k}(r) = r
  \]
- Containment in one direction always holds:
  \[
  \pi_{X_1}(r) \Join \cdots \Join \pi_{X_k}(r) \supseteq r
  \]
- What about the other direction? Depends on \( F \! \)!

Example 1

Example 2

Decision Algorithm

Losslessness Testing

Given:
- \( U, F, X_1, \ldots, X_k \)
- \( \{X_1, \ldots, X_k\} \) is a decomposition of \((U,F)\)

Determine whether \( \{X_1, \ldots, X_k\} \) is a lossless decomposition

- The definition of lossless is not constructive (check every possible relation)
- Next, we present a polynomial-time algorithm for this decision problem

The Case of Binary Decomposition

THEOREM: Let \( \{X_1, X_2\} \) be a decomposition of \((U,F)\). The following are equivalent:

1. \( F \not\models (X_1 \cap X_2) \rightarrow X_1 \) or \( F \not\models (X_1 \cap X_2) \rightarrow X_2 \)
2. \( \{X_1, X_2\} \) is a lossless decomposition

So what would be a decision algorithm in this case?
Proof: 1 $\Rightarrow$ 2

1. $F = X_1 \cap X_2 \rightarrow X_1$ or $F = X_1 \cap X_2 \rightarrow X_2$
2. $(X_1, X_2)$ is a lossless decomposition

Illustration: not 1 $\Rightarrow$ 2

1. $F = X_1 \cap X_2 \rightarrow X_1$ or $F = X_1 \cap X_2 \rightarrow X_2$
2. $(X_1, X_2)$ is a lossless decomposition

The Idea

1. $F = X_1 \cap X_2 \rightarrow X_1$ or $F = X_1 \cap X_2 \rightarrow X_2$
2. $(X_1, X_2)$ is a lossless decomposition

The General Case

Losslessness Testing

Given:
- $U, F, X_1, \ldots, X_k$
- $(X_1, \ldots, X_k)$ is a decomposition of $(U, F)$

Goal:
Determine whether $(X_1, \ldots, X_k)$ is a lossless decomposition

Next, we handle the general case of a decomposition ($\geq 2$ schemas)

Proof: not 1 $\Rightarrow$ not 2

1. $F = X_1 \cap X_2 \rightarrow X_1$ or $F = X_1 \cap X_2 \rightarrow X_2$
2. $(X_1, X_2)$ is a lossless decomposition

• Let $X_{12} = (X_1 \cap X_2)^+$ and assume $X_1 \not\subseteq X_{12}$, $X_2 \not\subseteq X_{12}$
• Construct a relation $r = \{1, 0\}$ over $U$:
  - $|X_{12}| = |U| \cap X_{12} = (0, 0)$
  - $|U \setminus X_{12}| = (1, 1)$
  - $|U \setminus X_{12}| = (2, 2)$
• Claim 1: $r \not\models F$
  - Proof similar to completeness of Armstrong’s axioms
• Claim 2: $\pi_{X_1}(r) \not\bowtie \pi_{X_2}(r) = r$
  - The join contains a row with both 1s and 2s

Negative Example

We need to prove that 1 is here!

We know that this is a subset of $r$, for some $x$'s

But some of the $x$'s may be known due to the FDs!

We need to prove that 1 is here!

Lossy! (how do we know that?)

We know that this is a subset of $r$, for some $x$'s

Stop!!
The General Case

<table>
<thead>
<tr>
<th>Given:</th>
<th>Goal:</th>
</tr>
</thead>
<tbody>
<tr>
<td>U,F,X₁,...,Xₖ</td>
<td>Determine whether (X₁,...,Xₖ) is a lossless decomposition</td>
</tr>
<tr>
<td>(X₁,...,Xₖ) is a decomposition of (U,F)</td>
<td></td>
</tr>
</tbody>
</table>

- 1st step: create the "known subset"
  - A table over U, one tuple tᵢ for each Xᵢ, tᵢ[Aⱼ]=aⱼ if Xᵢ contains Aⱼ and t[Aⱼ]=xⱼ otherwise
- 2nd step: chase
  - While the table changes do:
    - Look for an FD violation and equate the conclusions
    - "Equate" = change every occurrence of one to the other
  - When equating aᵢ with xᵢ, change xᵢ to aᵢ
- 3rd step: Return true if there is a row of aᵢ's

Think

How do we generalize the proof of correctness from the two-table case?

Why is this algorithm terminating in polynomial time?

Example

<table>
<thead>
<tr>
<th>F = {A₁→A₅, A₅→A₆, A₆→A₇A₈}</th>
</tr>
</thead>
</table>

Step 1: construct the known subset

Step 2: chase

Step 3: return true

Outline

- Introduction
- Normal Forms
  - BCNF
  - 3NF
- Decomposition
  - NF Decompositions
  - Preserving Data
  - Preserving Dependencies
- Decomposition Algorithms
  - 3NF
  - BCNF (complementary)
  - Note on 4NF (complementary)

Preserving Dependencies

(U,F)

<table>
<thead>
<tr>
<th>r</th>
<th>X₅F₁</th>
<th>X₆F₂</th>
<th>X₇F₃</th>
<th>X₈F₄</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Is F preserved given that each Fᵢ is preserved in each relation?

Example 1

<table>
<thead>
<tr>
<th>ABCD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A→B, B→C, ABC→D, D→B</td>
</tr>
</tbody>
</table>

{AD,BD,BC}

Are dependencies preserved in this decomposition?

Answer: Yes!
Testing for Dependency Preservation

Example

- We need to test whether $F^+ = (F_1 \cup \cdots \cup F_k)^+$
- $F^+ \supseteq F_1 \cup \cdots \cup F_k$, so $F^+ \supseteq (F_1 \cup \cdots \cup F_k)^+$
- So, need to test whether $F^+ \subseteq (F_1 \cup \cdots \cup F_k)^+$
- It suffices to test whether each $X \rightarrow Y$ in $F$ is implied by $F_1 \cup \cdots \cup F_k$
- Or in other words, whether $Y$ is a subset of the closure of $X$ under $F_1 \cup \cdots \cup F_k$
- Next slide: efficient computation of the closure of $X$ under $F_1 \cup \cdots \cup F_k$
- Without explicitly calculating the $F_i$'s!

Formal Definition

- A decomposition $X_1, \ldots, X_k$ of $(U, F)$ is dependency preserving if for all $r_1, \ldots, r_k$ over $(X_1, F_1), \ldots, (X_k, F_k)$, respectively, $F_i = F[X_i]$, the relation $r_1 \bigcirc \cdots \bigcirc r_k$ satisfies $F$
- Can we test whether a given decomposition has this property?
- Theorem: The following are equivalent:
  1. For all $r_1, \ldots, r_k$ over $(X_1, F_1), \ldots, (X_k, F_k)$, respectively, the relation $r_1 \bigcirc \cdots \bigcirc r_k$ satisfies $F$
  2. $F^* = (F_1 \cup \cdots \cup F_k)^+$

Closure w.r.t. a Decomposition

Basic claim for $Z \subseteq X_i$:
$Z_i \cap X_i = Z_i$

DepPreserving($X_1, \ldots, X_k, F$) {
  for all $(X \rightarrow Y$ in $F$)
    if $(Y \subseteq \text{ClosureDecomp}(X, F, X_1, \ldots, X_k))$
      return false
  return true
}

Outline

- Introduction
  - Normal Forms
    - BCNF
    - 3NF
  - Decomposition
    - NF Decompositions
    - Preserving Data
    - Preserving Dependencies
  - Decomposition Algorithms
    - 3NF
    - BCNF (complementary)
    - Note on 4NF (complementary)
Decomposition Algorithms

- Given a normal form \( N \), we ask:
  - Is there always a lossless \( N \) decomposition?
  - Is there always a lossless & dependency preserving \( N \) decomposition?
  - Is there an efficient decomposition?

- The next slides discuss two algorithms
  - 3NF decomposition
    - Lossless, dependency preserving, p-time
  - BCNF decomposition
    - Lossless
    - Complementary (not official course material)

Outline

- Introduction
- Normal Forms
  - BCNF
  - 3NF
- Decomposition
  - NF Decompositions
  - Preserving Data
  - Preserving Dependencies
- Decomposition Algorithms
  - 3NF
  - BCNF (complementary)
  - Note on 4NF (complementary)

Intuition

Idea: for dependency preservation, each \( X \rightarrow A \) becomes a schema

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

\( A \rightarrow B \) \( \rightarrow \) \( AB \rightarrow C \) \( \rightarrow \) \( BC \) \( \rightarrow \) \( CD \) \( \rightarrow \) \( AD \)

Problem: not in 3NF
Solution: minimal cover instead of \( F \)

Reminder: Minimal Cover

- Let \( F \) be a set of FDs
- A minimal cover of \( F \) is a set \( G \) of FDs such that \( G^+ = F^+ \) with the following properties:
  - FDs in \( G \) have a single attribute on the right hand side; that is, they have the form \( X \rightarrow A \)
  - All FDs are required: no FD \( X \rightarrow A \) in \( G \) is such that \( G \setminus \{X \rightarrow A\} \vDash X \rightarrow A \)
  - All attributes are required: no FD \( XB \rightarrow A \) in \( G \) is such that \( G \vDash X \rightarrow A \)

Revised Example

\{ \( A \rightarrow B \), \( AB \rightarrow C \), \( C \rightarrow B \), \( D \rightarrow C \) \}

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

\( A \rightarrow C \) \( \rightarrow \) \( BC \) \( \rightarrow \) \( CD \) \( \rightarrow \) \( AD \)

Algorithm for 3NF Decomposition

\[ 3NFDec(U, F) \}
\[ D = \emptyset \]
\[ G := \text{MinCover}(r) \]
\[ \text{for all } (X \rightarrow A \text{ in } G) \text{ do} \]
\[ D := DU(XA) \]
\[ \text{if (no set in } D \text{ is a superkey)} \]
\[ D := DU(\text{FindKey}(U, F)) \]
\[ D := \text{RemoveContained}(D) \]
\[ \text{return } D \]

No need for schemas contained in others
About the Proof

• We will not prove the correctness here
• Still, what needs to be proved?
  – Resulting schemas are all in 3NF
  • Follows from minimality of the cover
  – Dependencies are preserved
  • Straightforward: all dependencies of the minimal cover are presented
  – Lossless
  • What would the lossless-testing algorithm do when one Xᵢ is a key and each dependency is contained in one of the Xᵢs?

Example Revisited

OPTIONAL MATERIAL

The rest of the presentation is not in the official course material.

Outline

• Introduction
• Normal Forms
  • BCNF
  • 3NF
• Decomposition
  • NF Decompositions
  • Preserving Data
  • Preserving Dependencies
• Decomposition Algorithms
  • 3NF
  • BCNF (complementary)
  • Note on 4NF (complementary)

Key Insight

• Recall: BCNF means that in every nontrivial X → Y, the set X is a superkey
• CLAIM: If (U,F) is not in BCNF, then there is a lossless decomposition {X₁,X₂} with X₁,X₂ ⊆ U
• Proof:
  – Let X → Y be a BCNF violation (X is not a superkey and Y is not a subset of X)
  – Take X₁=X⁺ and X₂=X∪(U\X⁺)
  – Why are X₁ and X₂ strict subsets of U?
  – Why lossless?
  • Recall the theorem on binary lossless decompositions ...

BCNF Decomposition

```
BCNFDec(U,F) {
  if ((U,F) in BCNF)
    return {U}
  Find a BCNF violation X → Y
  X₁ := Closure(X,F)
  F₁ := F[X₁]
  X₂ := X U (U\X₁)
  F₂ := F[X₂]
  return BCNFDec(X₁,F₁) U BCNFDec(X₂,F₂)
}
```
### Execution Example

```
ABCD
A→B, B→C, ABC→D, D→B
```

```
BC
B→C
```

```
A\&D
```

```
A→B
```

```
BD
D→B
```

```
{AD, BD, BC}
```

Are dependencies preserved in this decomposition?

Answer: Yes, we already saw that previously.

### About the Algorithm

- **Lossless**
  - Proof idea: every step is lossless
- **Exponential time** in the worst case
- There is a polynomial-time algorithm for BCNF decomposition
  - [Tsou, Fischer, Decomposition of a relation scheme into Boyce-Codd Normal Form, 1982]
- The algorithm does not preserve dependencies!
  - But the problem is not with the algorithm...

### Can Dependencies be Preserved?

```
ABC
AB→C, C→B
```

```
BC
C→B
```

```
AC
```

No BCNF decomposition of this schema preserves both dependencies (why?)

Conclusion: Lossless BCNF decomposition is always possible; lossless & dependency-preserving BCNF decomposition may be impossible.

### Outline

- Introduction
- Normal Forms
  - BCNF
  - 3NF
- Decomposition
  - NF Decompositions
  - Preserving Data
  - Preserving Dependencies
- Decomposition Algorithms
  - 3NF
  - BCNF (complementary)
  - Note on 4NF (complementary)

### Fourth Normal Form (4NF)

- Recall: An MVD has the form $X\rightarrow Y$ where $X$ and $Y$ are disjoint sets of attributes
  - For every two tuples that agree on $X$, swapping their $Y$ component doesn’t change the relation
- Recall: An MVD $X\rightarrow Y$ is trivial (always holds) if and only if $Y=\emptyset$ or $Y=U\setminus X$
- Recall: an FD $X\rightarrow Y$ can be viewed as a special type of the MVD $X\rightarrow Y$ (why?)
- A schema $(U,F)$, where $F$ contains both FDs and MVDs, is in 4NF if every nontrivial FD/MVD has a superkey in its premise (LHS)
  - When all dependencies are FDs, same as BCNF

### 4NF Decomposition

- **THEOREM**: Let $(U,F)$ be a schema, where $F$ contains both FDs and MVDs. Then $F$ satisfies $X\rightarrow Y$ if for all relations $r$ over $U$ we have:
  $$ r = \pi_X(f) \gg \pi_{U\setminus X}(f) $$
- Hence, the recursive decomposition algorithm for BCNF decomposition works here
  - Decompose$(XU\setminus Y) \cup \text{Decompose}(XU\setminus Y))$
  - A polynomial time is known for special cases
- In particular, there is always a lossless 4NF decomposition
  - What about dependency preserving?
  - Answer: No! Even if there are only FDs (recall BCNF)