# COSI 127b
# Introduction to Database Systems

## Lecture 15:  Normalization (4)

# Review: Good DB Design

## Three Approaches:

1. Ad hoc:
   - use Entity-Relationship Model to model data requirements
   - translate ER design into relational schema

   Issue: How to tell if design is "good"?

2. Theoretical:
   - construct universal relations (e.g., `Borrower-All`)
   - decompose above using known functional dependencies

   Issue: Time-Consuming and Complex

3. Practical:
   - use ER Model to produce 1$^{st}$ cut DB design
   - use FDs to refine and verify

# Review: Functional Dependencies

## Previously:

- What "$A_1, \ldots, A_n \rightarrow B$" means
- When sets of FDs are equivalent ($F \equiv G$)
  - *if* $F^+ = G^+$ *(FD set closures)*
  - *algorithms: Attribute Closures or Armstrong's Axioms*
- Minimal FD Sets ($F_c$ = "Canonical Cover" of F)
- Canonical Cover Algorithm

## Today:

- DB Design using FDs

# Review: Canonical Cover ($F_C$)

## One more algorithm over FD sets:

- Canonical Cover ($F_C$): a "minimal" version of FD set, $F$
- $F_C$ the "minimal" version of $F$?
  1. equivalent to $F$ ($F_C^+ = F^+$)
  2. "smaller" than other FD sets equivalent to $F$:
     a) fewer FDs:
        $\{A \to B,\ B \to C\} < \{A \to B,\ B \to C,\ A \to C\}$
     b) fewer attributes in FDs:
        $\{A \to B,\ B \to C\} < \{A \to BC,\ B \to C\}$

# Review: Canonical Cover ($F_C$)

## Canonical Cover Algorithm

```
ALGORITHM Canonical-Cover (F: {FDs})
BEGIN
 REPEAT UNTIL STABLE
  1. Where possible, apply UNION rule to FD's in F
     (Armstrong's Axioms)

  2. Remove extraneous attributes from each FD in F

    a) RHS:  Is B extraneous in A → BC?

             Is (A → B) ∈ (F − {A → BC} ∪ {A → C})+?

    b) LHS:  Is B extraneous in AB → C?

             Is (A → C) ∈ F+ ?
END
```

# Review: Normalization

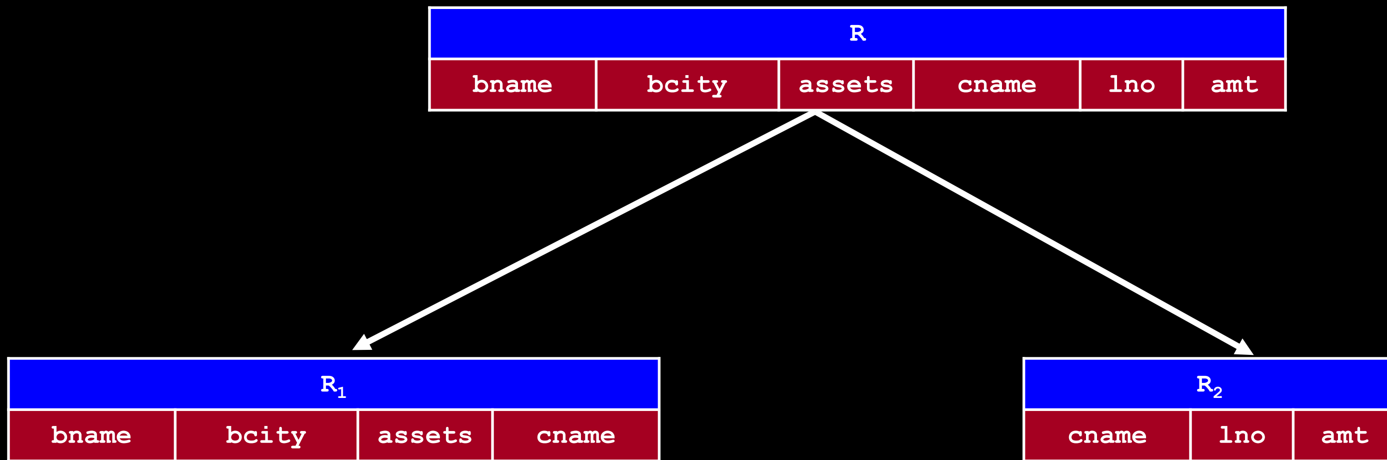
Basic Idea:

1. Start with Universal Relation(s), R
   - all attributes in 1-2 tables
   - e.g., Borrower-All, Depositor-All

| Borrower-All | | | | | | | |
|---|---|---|---|---|---|---|---|
| **lno** | **cname** | **cstreet** | **ccity** | **bname** | **amt** | **bcity** | **assets** |
| L-17 | Jones | Main | Harrison | Downtown | 1000 | Brooklyn | 9M |
| L-23 | Smith | North | Rye | Redwood | 2000 | Palo Alto | 2.1M |
| L-15 | Hayes | Main | Harrison | Perry | 1500 | Horseneck | 1.7M |
| L-17 | Jackson | Senator | Brooklyn | Downtown | 1000 | Brooklyn | 9M |
| L-93 | Curry | Walnut | Stanford | Mianus | 500 | Horseneck | 0.4M |
| L-11 | Smith | North | Rye | R.H. | 900 | Horseneck | 8M |
| L-16 | Adams | Spring | Pittsfield | Perry | 1300 | Horseneck | 1.7M |

2. Determine FD set for R, F
3. Decompose R according to FDs in $F^+$

# Review: Decomposition




Notation for schema decomposition:

$R = R_1 \cup R_2$

*BTW: Not a Good Decomposition*

# Review: Decomposition Goals

1. Lossless Joins
   - Avoid information loss

2. Redundancy Avoidance
   - Avoid update anomalies

*Relative Importance:*
*1: Primary Importance*
*2,3: Secondary Importance*

3. Dependency Preservation
   - Avoid expensive global integrity constraints

# Review: Decomposition Goal Tests

Test of $R = R_1 \cup \ldots \cup R_n$ with FD set $F$:

- Lossless Joins? iff for each decomposition step, $R_i = R_i \cup R_j$:

  $(\mathbf{R_1} \cap \mathbf{R_2} \rightarrow \mathbf{R_1})$ or $(\mathbf{R_1} \cap \mathbf{R_2} \rightarrow \mathbf{R_2})$

# Review: Lossless Joins Test

Example 1: $R = R_1 \cup R_2$ ($R$ = (A,B,C))

$R_1$ = (A,C)
$R_2$ = (B,C)
F = {**AB** → **C**, **C** → **B**}

**Is the decomposition of R lossless?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

**AC** → $\mathbf{R_1}$
**C** → $\mathbf{R_2}$

2) What is $R_1 \cap R_2$?

C

3) Does $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$ ?

Yes, **C** → $\mathbf{R_2}$

Therefore, decomposition of R is **lossless**

# Review: Lossless Joins Test

Example 2: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,B,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Is the decomposition of R lossless?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AB \rightarrow R_1$, $AC \rightarrow R_1$
$C \rightarrow R_2$

2) What is $R_1 \cap R_2$?

BC

3) Does $R_1 \cap R_2 \rightarrow R_1$ or $R_1 \cap R_2 \rightarrow R_2$ ?

Yes. $BC \rightarrow R_2$

Therefore, decomposition of R is **lossless**

# Review: Decomposition Goal Tests

Test of $R = R_1 \cup \ldots \cup R_n$ with FD set $F$:

- Lossless Joins? iff for each decomposition step, $R_i = R_i \cup R_j$:

  $(\mathbf{R_1} \cap \mathbf{R_2} \rightarrow \mathbf{R_1})$ or $(\mathbf{R_1} \cap \mathbf{R_2} \rightarrow \mathbf{R_2})$

- Redundancy Avoidance? iff for each $R_i$ in decomposition result:

  for each nontrivial, $\mathbf{X} \rightarrow \mathbf{Y}$ in $F^+$ covered by $R_i$, $\mathbf{X} \rightarrow \mathbf{R_i}$

# Review: Redundancy Avoidance Test

Example 1: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Does decomposition of R have redundancy?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AC \rightarrow R_1$
$C \rightarrow R_2$

2) Which non-trivial FDs of $F^+$ are covered by $R_1$?

– ✓

3) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$ $(C \rightarrow R_2)$ ✓

Therefore, decomposition of R **<u>has no</u>** redundancy

# Review: Redundancy Avoidance Test

Example 2: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,B,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Does decomposition of R have redundancy?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AB \rightarrow R_1$, $AC \rightarrow R_1$
$C \rightarrow R_2$

2) Which non-trivial FDs of $F^+$ are covered by $R_1$?

$AB \rightarrow C$ $(AB \rightarrow R_1)$
$C \rightarrow B$ $(C \not\rightarrow A$ so $C \not\rightarrow R_1)$

3) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$ $(C \rightarrow R_2)$

Therefore, decomposition of R **has** redundancy

# Review: Decomposition Goal Tests

Test of $R = R_1 \cup \ldots \cup R_n$ with FD set $F$:

- Lossless Joins? iff for each decomposition step, $R_i = R_i \cup R_j$:

  $(R_1 \cap R_2 \rightarrow R_1)$ or $(R_1 \cap R_2 \rightarrow R_2)$

- Redundancy Avoidance? iff for each $R_i$ in decomposition result:

  for each nontrivial, $X \rightarrow Y$ in $F^+$ covered by $R_i$, $X \rightarrow R_i$

- Dependency Preserving? iff:

$$\left( \bigcup_{i=1}^{n} \{ f \in F^+ \mid f \text{ covered by } R_i \} \right) = F^+$$

# Review: Dependency Preservation Test

Example 1: $R = R_1 \cup R_2$ ($R = (A,B,C)$)

$R_1 = (A,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Is decomposition of R dependency preserving?**

A: 1) Which non-trivial FDs of $F^+$ are covered by $R_1$?

–

2) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$

3) Does $(1 \cup 2)^+ = F^+$?

No. $(AB \rightarrow C) \in F^+$ but $(AB \rightarrow C) \notin \{C \rightarrow B\}^+$

Therefore, decomposition of R **is not** dependency preserving

# Review: Dependency Preservation Test

Example 2: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,B,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Is decomposition of R dependency preserving?**

A: 1) Which non-trivial FDs of $F^+$ are covered by $R_1$?

$AB \rightarrow C$
$C \rightarrow B$

2) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$

3) Does $(1 \cup 2)^+ = F^+$?

Yes. $\{AB \rightarrow C,\ C \rightarrow B\}^+ = F^+$

**Therefore, decomposition of R <u>is</u> dependency preserving**

# Review: Tests of Decomposition Goals

| Example | Lossless Joins? | Avoids Redundancy? | Dependency Preserving? |
|---|---|---|---|
| $R_1 = (A, C)$<br>$R_2 = (B, C)$<br>$F = \{AB \rightarrow C,\ C \rightarrow B\}$ | yes | yes | no |
| $R_1 = (A, B, C)$<br>$R_2 = (B, C)$<br>$F = \{AB \rightarrow C,\ C \rightarrow B\}$ | yes | no | yes |

# Review: Goals of Decomposition

| Goal | Motivation | Idea | Test |
|---|---|---|---|
| Lossless Joins | avoid info loss | recomposing tables should not add noise | For: $R = R_1 \cup R_2$<br>$(R_1 \cap R_2) \rightarrow R_1$ or<br>$(R_1 \cap R_2) \rightarrow R_2$ |
| Redundancy Avoidance | avoid update and deletion anomalies | only FD's with keys covered by decomposed tables | For any $X \rightarrow Y$ covered by $R_i$, $X$ is a superkey of $R_i$ |
| Dependency Preservation | efficient FD enforcement | fewer global ICs required to enforce FDs | For: $R = R_1 \cup \ldots \cup R_n$<br>(FD's covered by each $R_i$)$^+$<br>=<br>(FD's covered by $R$)$^+$ |

# Goals of Decomposition

| Goal | Motivation | Idea | Test | Guaranteed By |
|---|---|---|---|---|
| Lossless Joins | avoid info loss | recomposing tables should not add noise | For: $R = R_1 \cup R_2$ <br> $(R_1 \cap R_2) \to R_1$ or <br> $(R_1 \cap R_2) \to R_2$ | **BCNF, 3NF** |
| Redundancy Avoidance | avoid update and deletion anomalies | only FD's with keys covered by decomposed tables | For any $X \to Y$ covered by $R_i$, $X$ is a superkey of $R_i$ | **BCNF** |
| Dependency Preservation | efficient FD enforcement | fewer global ICs required to enforce FDs | For: $R = R_1 \cup \ldots \cup R_n$ <br> (FD's covered by each $R_i$)$^+$ <br> = <br> (FD's covered by $R$)$^+$ | **3NF** |

# Normalization

## Normal Forms

- schema in some normal form if it satisfies certain properties

  e.g., BCNF: lhs of FD covered by table is a key

- typically accompanied by decomposition algorithm that ensures result schema satisfies normal form

# Normalization

## Boyce-Codd Normal Form (BCNF):

- can result in many decompositions (schemas) for same relation + FD Set
- all result schemas satisfy

  1. lossless joins
  2. redundancy avoidance
  3. dependency preservation (sometimes, but not always possible)

## Third Normal Form (3NF):

- can result in many decompositions (schemas) for same relation + FD Set
- all result schemas satisfy

  1. lossless joins
  2. dependency preservation (at least one schema satisfies)
  3. redundancy avoidance (sometimes, but not always possible)

# Boyce-Codd Normal Form (BCNF)

## Boyce-Codd Normal Form (BCNF):

- can result in many decompositions (schemas) for same relation + FD Set
- all result schemas satisfy

1. lossless joins
2. redundancy avoidance
3. dependency preservation (sometimes, but not always possible)

## Informally:

Relation schema R, with FD set F, is in BCNF if it avoids redundancy

Decomposition $R = R_1 \cup \ldots \cup R_n$ with FD set F, is in BCNF if every resulting relation, $R_i$, is in BCNF

# Boyce-Codd Normal Form (BCNF)

## Boyce-Codd Normal Form (BCNF):

- can result in many decompositions (schemas) for same relation + FD Set
- all result schemas satisfy

1. lossless joins
2. redundancy avoidance
3. dependency preservation (sometimes, but not always possible)

## Formally:

Relation schema R, with FD set F, is in BCNF if

for every nontrivial FD, $X \rightarrow Y$ in $F^+$ that is covered by R, $X \rightarrow R$

Decomposition $R = R_1 \cup \ldots \cup R_n$ with FD set F, is in BCNF if every resulting relation, $R_i$, is in BCNF

# Review: Redundancy Avoidance Test

Example 1: $R = R_1 \cup R_2$ ($R = (A,B,C)$)

$R_1 = (A,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Does decomposition of R have redundancy?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AC \rightarrow R_1$
$C \rightarrow R_2$

2) Which non-trivial FDs of $F^+$ are covered by $R_1$?

– ✓

3) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$ ($C \rightarrow R_2$) ✓

Therefore, decomposition of R **has no** redundancy

# Test for BCNF

Example 1: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Is decomposition of R in BCNF?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AC \rightarrow R_1$
$C \rightarrow R_2$

2) Which non-trivial FDs of $F^+$ are covered by $R_1$?

–  ✓

3) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$   $(C \rightarrow R_2)$ ✓

**Therefore, decomposition of R <u>is</u> in BCNF**

# Review: Redundancy Avoidance Test

Example 2: $R = R_1 \cup R_2$ ($R = (A,B,C)$)

$R_1 = (A,B,C)$
$R_2 = (B,C)$
$F = \{AB \to C,\ C \to B\}$

**Does decomposition of R have redundancy?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AB \to R_1$, $AC \to R_1$
$C \to R_2$

2) Which non-trivial FDs of $F^+$ are covered by $R_1$?

$AB \to C$ ($AB \to R_1$) ? ✓
$C \to B$ ($C \not\to A$ so $C \not\to R_1$) ✗

3) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \to B$ ($C \to R_2$) ? ✓

Therefore, decomposition of R **has** redundancy

# Test for BCNF

Example 2: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,B,C)$
$R_2 = (B,C)$
$F = \{AB \rightarrow C,\ C \rightarrow B\}$

**Is decomposition of R in BCNF?**

A: 1) What are the candidate keys of $R_1$, $R_2$?

$AB \rightarrow R_1$, $AC \rightarrow R_1$
$C \rightarrow R_2$

2) Which non-trivial FDs of $F^+$ are covered by $R_1$?

$AB \rightarrow C$ $(AB \rightarrow R_1)$ ?
$C \rightarrow B$ $(C \not\rightarrow A$ so $C \not\rightarrow R_1)$

3) Which non-trivial FDs of $F^+$ are covered by $R_2$?

$C \rightarrow B$ $(C \rightarrow R_2)$ ?

Therefore, decomposition of R **is not** in BCNF

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*BCNF Decomposition Algorithm*

*Input: R: Relation, F: FD set*
*Output: Set of Relations forming decomposition of R*

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*BCNF Decomposition Algorithm*

*Input: R: Relation, F: FD set*
*Output: Set of Relations forming decomposition of R*

*Initial result: {R}*

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*BCNF Decomposition Algorithm*

*Input: R: Relation, F: FD set*
*Output: Set of Relations forming decomposition of R*

*Initial result: {R}*

*Repeat until all tables in result in BCNF*

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*BCNF Decomposition Algorithm*

*Input: R: Relation, F: FD set*
*Output: Set of Relations forming decomposition of R*

*Initial result: {R}*

*Repeat until all tables in result in BCNF*

*Pick an* $R_i$ *not in BCNF*

*Decompose on* $R_i$

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*BCNF Decomposition Algorithm*

*Input: R: Relation, F: FD set*
*Output: Set of Relations forming decomposition of R*

*Initial result: {R}*

*Repeat until all tables in result in BCNF*

*Pick an $R_i$ not in BCNF*
*i.e., some FD,($X \rightarrow Y$) $\in F^+$ covered by $R_i$ where X not a key of $R_i$*

*Decompose on $R_i$*

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*BCNF Decomposition Algorithm*

*Input: R: Relation, F: FD set*
*Output: Set of Relations forming decomposition of R*

*Initial result: {R}*

*Repeat until all tables in result in BCNF*

*Pick an $R_i$ not in BCNF*
*i.e., some FD,$(\mathbf{X} \rightarrow \mathbf{Y}) \in F^+$ covered by $R_i$ where X not a key of $R_i$*

*Decompose on $R_i$*
*i.e., replace $R_i$ with $\{R_{i1} = X \cup Y, R_{i2} = R_i - Y\}$ in result*

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

*Intuition: At each step until decomposition is in BCNF…*

Decompose $R_i$ on $(X \rightarrow Y)$ ($X$ *not a key of* $R_i$)

| $R_i$ | | | | | |
|---|---|---|---|---|---|
| A | B | C | D | E | H |

*decompose* $R_i$ *on* $(B \rightarrow CD)$

$R_{i1} \leftarrow X \cup Y$

| $R_{i1}$ | | |
|---|---|---|
| B | C | D |

| $R_{i2}$ | | | |
|---|---|---|---|
| A | B | E | H |

$R_{i2} \leftarrow R_i - Y$

*Observe:* **B** $\rightarrow$ **CD** *covered and* **B** $\rightarrow$ $\mathbf{R_{i1}}$
*Therefore,* $R_{i1}$ *in BCNF*

$R_{i1}$ *may not be in BCNF, but we have made progress! (***B** $\rightarrow$ **CD** *not covered)*

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
   Compute F+
   Result ← {R}
   WHILE some Ri ∈ Result not in BCNF DO
     Choose non-trivial (X → Y) ∈ F+ such that:
       • X not a key of Ri, and
       • (X → Y) covered by Ri
     Decompose Ri on (X → Y)
       • Ri1 ← X ∪ Y
       • Ri2 ← Ri – Y
     Result ← Result – {Ri} ∪ {Ri1,Ri2}
   RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}

| R | | |
|---|---|---|
| A | B | C |

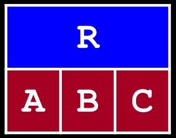
```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri - Y
      Result ← Result - {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}




```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri – Y
      Result ← Result – {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

| R | | |
|---|---|---|
| A | B | C |

*What's the (Candidate) Key of* R*?*

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri - Y
      Result ← Result - {Ri} ∪ {Ri1,Ri2}
    RETURN Result

END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}

$F^+$ = {**A** → **B**, **B** → **C**, **A** → **C**, …}

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri - Y
      Result ← Result - {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

| R | | |
|---|---|---|
| <u>A</u> | B | C |

*What's the (Candidate) Key of* R*?*

A: A

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}

$F^+$ = {**A** → **B**, **B** → **C**, **A** → **C**, …}

| R | | |
|---|---|---|
| A | B | C |

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
           •   (X → Y) covered by Ri
           •    X not a key of Ri
      Decompose Ri on (X → Y)
           •   Ri1 ← X ∪ Y
           •   Ri2 ← Ri − Y
      Result ← Result − {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

| R | | |
|---|---|---|
| <u>A</u> | B | C |

**ALGORITHM BCNF** (R: Relation, F: FD set)

**BEGIN**

Compute $F^+$

**Result $\leftarrow$ {R}**

**WHILE** some $R_i \in$ Result not in BCNF **DO**

Choose non-trivial $(X \rightarrow Y) \in F^+$ such that:

- $(X \rightarrow Y)$ covered by $R_i$
- X not a key of $R_i$

Decompose $R_i$ on $(X \rightarrow Y)$

- $R_{i1} \leftarrow X \cup Y$
- $R_{i2} \leftarrow R_i - Y$

Result $\leftarrow$ Result $- \{R_i\} \cup \{R_{i1}, R_{i2}\}$

**RETURN** Result

**END**

**Result = {R}**

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

| R | | |
|---|---|---|
| <u>A</u> | B | C |

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some R_i ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
           •   (X → Y) covered by R_i
           •    X not a key of R_i
      Decompose R_i on (X → Y)
           •    R_i1 ← X ∪ Y
           •    R_i2 ← R_i - Y
      Result ← Result - {R_i} ∪ {R_i1,R_i2}
    RETURN Result
END
```

**Result = {R}**

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

R *not in BCNF because of* **B** → **C**

| R | | |
|---|---|---|
| <u>A</u> | B | C |

**ALGORITHM BCNF** (R: Relation, F: FD set)
**BEGIN**
Compute $F^+$
Result ← {R}
**WHILE some $R_i \in$ Result not in BCNF DO**
**Choose non-trivial $(X \rightarrow Y) \in F^+$ such that:**
- **$(X \rightarrow Y)$ covered by $R_i$**
- **X not a key of $R_i$**

Decompose $R_i$ on (**X** → **Y**)
- $R_{i1} \leftarrow X \cup Y$
- $R_{i2} \leftarrow R_i - Y$

Result ← Result − $\{R_i\} \cup \{R_{i1}, R_{i2}\}$
**RETURN** Result
**END**

**Result = {R}**

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}

$F^+$ = {**A** → **B**, **B** → **C**, **A** → **C**, …}

R *not in BCNF because of* **B** → **C**

| R | | |
|---|---|---|
| <u>A</u> | B | C |

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
         •   (X → Y) covered by Ri
         •    X not a key of Ri
      Decompose Ri on (X → Y)
         •   Ri1 ← X ∪ Y
         •   Ri2 ← Ri − Y
      Result ← Result − {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

**Result = {R}**

# Boyce-Codd Normal Form (BCNF)

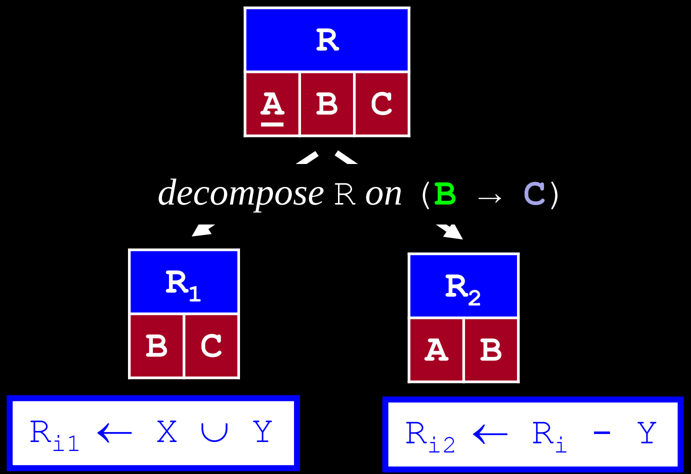
## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}

$F^+$ = {**A** → **B**, **B** → **C**, **A** → **C**, ...}

R *not in BCNF because of* **B** → **C**




```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri - Y
      Result ← Result - {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

**Result = {R}**

# Boyce-Codd Normal Form (BCNF)

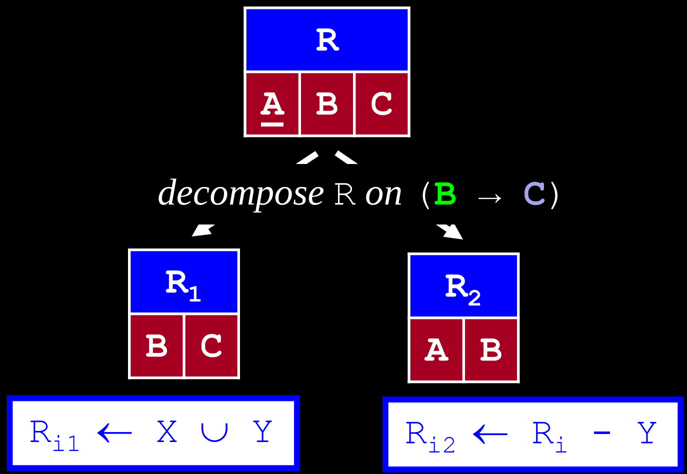
## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B, B \rightarrow C\}$

$F^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, \ldots\}$

R *not in BCNF because of* $B \rightarrow C$




**Result = {R}**

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri - Y
      Result ← Result - {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

$R$ *not in BCNF because of* $B \rightarrow C$

| R | | |
|---|---|---|
| <u>A</u> | B | C |

*decompose* $R$ *on* $(B \rightarrow C)$

| $R_1$ | |
|---|---|
| B | C |

| $R_2$ | |
|---|---|
| A | B |

$R_{i1} \leftarrow X \cup Y$

$R_{i2} \leftarrow R_i - Y$

**Result = $\{R_1, R_2\}$**

**ALGORITHM BCNF** (R: Relation, F: FD set)
**BEGIN**
  Compute $F^+$
  Result $\leftarrow \{R\}$
  **WHILE** some $R_i \in$ Result not in BCNF **DO**
    Choose non-trivial $(X \rightarrow Y) \in F^+$ such that:
    - $(X \rightarrow Y)$ covered by $R_i$
    - X not a key of $R_i$

    Decompose $R_i$ on $(X \rightarrow Y)$
    - $R_{i1} \leftarrow X \cup Y$
    - $R_{i2} \leftarrow R_i - Y$

    **Result $\leftarrow$ Result $- \{R_i\} \cup \{R_{i1}, R_{i2}\}$**
  **RETURN** Result
**END**

# Boyce-Codd Normal Form (BCNF)

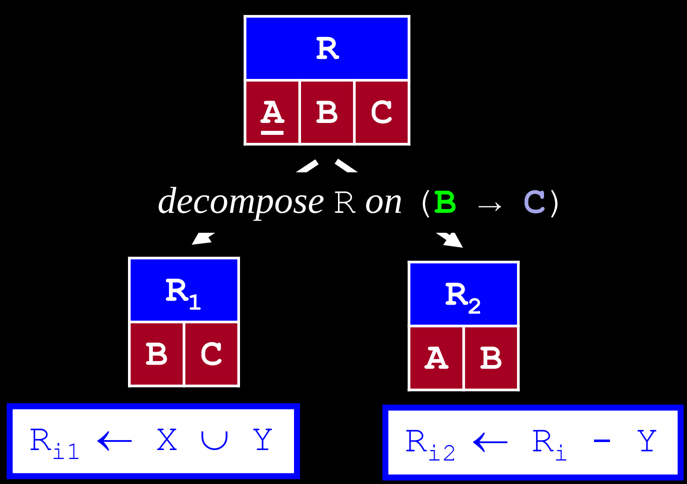
## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

R *not in BCNF because of* **B → C**




**Result = {R1, R2}**

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
           •   (X → Y) covered by Ri
           •    X not a key of Ri
      Decompose Ri on (X → Y)
           •   Ri1 ← X ∪ Y
           •   Ri2 ← Ri − Y
      Result ← Result − {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

**ALGORITHM BCNF** (R: Relation, F: FD set)
**BEGIN**
Compute $F^+$
Result $\leftarrow$ {R}
**WHILE some $R_i \in$ Result not in BCNF DO**
**Choose non-trivial $(X \rightarrow Y) \in F^+$ such that:**
- **$(X \rightarrow Y)$ covered by $R_i$**
- **X not a key of $R_i$**

Decompose $R_i$ on $(X \rightarrow Y)$
- $R_{i1} \leftarrow X \cup Y$
- $R_{i2} \leftarrow R_i - Y$

Result $\leftarrow$ Result $- \{R_i\} \cup \{R_{i1}, R_{i2}\}$
**RETURN** Result
**END**

| $R_1$ | |
|---|---|
| B | C |

| $R_2$ | |
|---|---|
| A | B |

- $R_1$ *only covers* $B \rightarrow C$
- B *a key of* $R_1$
- *therefore in BCNF*

- $R_2$ *only covers* $A \rightarrow B$
- A *a key of* $R_1$
- *therefore in BCNF*

**Result = $\{R_1, R_2\}$**

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
    Compute F+
    Result ← {R}
    WHILE some Ri ∈ Result not in BCNF DO
      Choose non-trivial (X → Y) ∈ F+ such that:
          •   (X → Y) covered by Ri
          •    X not a key of Ri
      Decompose Ri on (X → Y)
          •   Ri1 ← X ∪ Y
          •   Ri2 ← Ri - Y
      Result ← Result - {Ri} ∪ {Ri1,Ri2}
    RETURN Result
END
```

| $R_1$ | |
|---|---|
| <u>B</u> | C |

| $R_2$ | |
|---|---|
| <u>A</u> | B |

- $R_1$ *only covers* **B** → **C**
- B *a key of* $R_1$
- *therefore in BCNF*

- $R_2$ *only covers* **A** → **B**
- A *a key of* $R_1$
- *therefore in BCNF*

**Result = $\{R_1, R_2\}$**

# Boyce-Codd Normal Form (BCNF)

## A Simple Example:

R = (A, B, C)

F = {**A** → **B**, **B** → **C**}

$F^+$ = {**A** → **B**, **B** → **C**, **A** → **C**, …}

**Result = {$R_1$, $R_2$}**

| $R_1$ | |
|---|---|
| **B** | **C** |

| $R_2$ | |
|---|---|
| **A** | **B** |

**ALGORITHM BCNF** (R: Relation, F: FD set)

**BEGIN**

Compute $F^+$

Result ← {R}

**WHILE** some $R_i$ ∈ Result not in BCNF **DO**

Choose non-trivial (**X** → **Y**) ∈ $F^+$ such that:

- (**X** → **Y**) covered by $R_i$
- X not a key of $R_i$

Decompose $R_i$ on (**X** → **Y**)

- $R_{i1}$ ← X ∪ Y
- $R_{i2}$ ← $R_i$ − Y

Result ← Result − {$R_i$} ∪ {$R_{i1}$, $R_{i2}$}

**RETURN Result**

**END**

# Boyce-Codd Normal Form (BCNF)

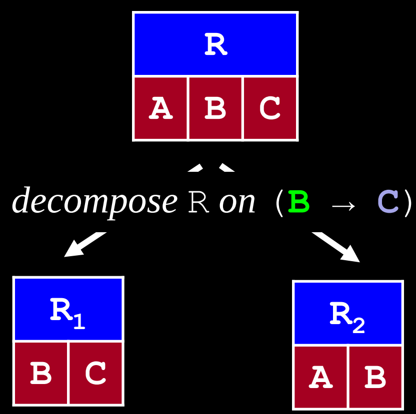
## How derivations are usually expressed:

$R = (A, B, C)$

$F = \{A \rightarrow B,\ B \rightarrow C\}$

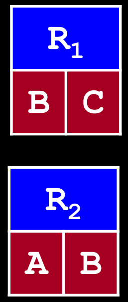
$F^+ = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C, \ldots\}$

*1. Derivation*




*2. Result*

$R = R_1 \cup R_2$



# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
   Compute F+
   Result ← {R}
   WHILE some Ri ∈ Result not in BCNF DO
     Choose non-trivial (X → Y) ∈ F+ such that:
       • X not a key of Ri, and
       • (X → Y) covered by Ri
     Decompose Ri on (X → Y)
       • Ri1 ← X ∪ Y
       • Ri2 ← Ri - Y
     Result ← Result - {Ri} ∪ {Ri1,Ri2}
   RETURN Result
END
```

# Boyce-Codd Normal Form (BCNF)

## An Algorithm to Decompose A Relation into BCNF:

**ALGORITHM BCNF** (R: Relation, F: FD set)

**BEGIN**

  **Compute $F^+$**

  Result $\leftarrow$ {

  **WHILE** some $R_i \in$ Result not in BCNF **DO**

    Choose non-trivial $(X \rightarrow Y) \in F^+$ such that:

    - $(X \rightarrow Y)$ covered by $R_i$
    - $X$ not a key of $R_i$

    Decompose $R_i$ on $(X \rightarrow Y)$

    - $R_{i1} \leftarrow X \cup Y$
    - $R_{i2} \leftarrow R_i - Y$

    Result $\leftarrow$ Result $- \{R_i\} \cup \{R_{i1}, R_{i2}\}$

  **RETURN** Result

**END**

*exponential cost (using either Attribute Closures, Armstrong's Axioms)!!*

# When an Algorithm Calls for $F^+$

## Where We Have Seen It…:

1) Equivalence of FD sets ($F^+ = G^+$)
   - **avoid** by using canonical cover algorithm

2) Canonical Cover Algorithm

```
… 2. Remove extraneous attributes from each FD in X

    a) RHS:  B extraneous in A → BC?
             True if (A → B) ∈ (F − {A → BC} ∪ {A → C})+ …

    b) LHS:  B extraneous in AB → C?
             True if (A → C) ∈ F+ …
```

   - **avoid** by computing attribute closure ($A^+$)

3) BCNF algorithm
   - **avoid** by …?

# When an Algorithm Calls for $F^+$

## Strategy: Compute $F^+$ Lazily…:

1) Compute $F_c$ using Canonical Cover Algorithm
2) Use $F_c$ to derive FD's in $F^+$ as needed, using Armstrong's Axioms

*official:*

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
   Compute F+
   Result ← {R}
   WHILE some Ri ∈ Result not in BCNF DO
     Choose non-trivial (X → Y) ∈ F+ such that:
       •  (X → Y) covered by Ri
       •   X not a key of Ri
     Decompose Ri on (X → Y)
       •  Ri1 ← X ∪ Y
       •  Ri2 ← Ri - Y
     Result ← Result - {Ri} ∪ {Ri1,Ri2}
   RETURN Result
END
```

# When an Algorithm Calls for $F^+$

## Strategy: Compute $F^+$ Lazily…:

1) Compute $F_c$ using Canonical Cover Algorithm
2) Use $F_c$ to derive FD's in $F^+$ as needed, using Armstrong's Axioms

*revised:*

```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result ← {R}
   WHILE some Ri ∈ Result not in BCNF DO
     Use Fc to derive non-trivial (X → Y) ∈ F+ s.t:
       •  (X → Y) covered by Ri
       •   X not a key of Ri
     Decompose Ri on (X → Y)
       •  Ri1 ← X ∪ Y
       •  Ri2 ← Ri - Y
     Result ← Result - {Ri} ∪ {Ri1,Ri2}
   RETURN Result
END
```

# When an Algorithm Calls for $F^+$

Example:

$R = (A, B, C, D, E)$

$F = \{\mathbf{A} \rightarrow \mathbf{B},\ \mathbf{C} \rightarrow \mathbf{B},\ \mathbf{BC} \rightarrow \mathbf{D}\}$

$\boxed{F_c = ?}$

1. B *extraneous in* **BC** → **D**? *Yes:* $(\mathbf{C} \rightarrow \mathbf{D}) \in F^+$

   Proof: $C^+ = \{C, B, D\}$. Therefore, $D \in B^+$

$F_c = \{\mathbf{A} \rightarrow \mathbf{B},\ \mathbf{C} \rightarrow \mathbf{BD}\}$

# Boyce-Codd Normal Form (BCNF)

**Example 1:**

R = (A, B, C, D)
F = {**A** → **B**, **AB** → **D**, **B** → **C**}

***Decompose* R *into BCNF***

*Compute* $F_c$: {**A** → **BD**, **B** → **C**}

B *extraneous in* **AB** → **D**? *Yes:* (**A** → **D**) $\in F^+$

Proof: $A^+ = \{A,B,C,D\}$. Therefore, $D \in A^+$

# Boyce-Codd Normal Form (BCNF)

## Example 1:

$R = (A, B, C, D)$
$F_c = \{A \rightarrow BD, B \rightarrow C\}$

***Decompose*** R ***into BCNF***

*Derivation:*

**R**

| A | B | C | D |
|---|---|---|---|

*decompose* R *on* ($B \rightarrow C$)

$R_{i1} \leftarrow X \cup Y$

**$R_1$**

| B | C |
|---|---|

- $R_1$ *only covers* $B \rightarrow C$
- B *a key of* $R_1$
- *therefore in BCNF*

**$R_2$**

| A | B | D |
|---|---|---|

$R_{i2} \leftarrow R_i - Y$

- $R_2$ *only covers* $A \rightarrow BD$, $A \rightarrow B$, $A \rightarrow D$
- A *a key of* $R_1$
- *therefore in BCNF*

# Boyce-Codd Normal Form (BCNF)

Example 1:

$R = (A, B, C, D)$
$F_c = \{A \rightarrow BD, \; B \rightarrow C\}$

*Decompose* R *into BCNF*

$R = R_1 \cup R_2$

| $R_1$ | |
| --- | --- |
| B | C |

| $R_2$ | | |
| --- | --- | --- |
| A | B | D |

$$\left.\begin{array}{c} \textit{covers } B \rightarrow C \\ \cup \\ \textit{covers } A \rightarrow BD \end{array}\right\} \Rightarrow \textit{covers } F_c \Rightarrow \textit{covers } F^+$$

*DP or not DP?*

A: *DP*

# Boyce-Codd Normal Form (BCNF)

Example 2:

$R = (A, B, C, D, E, H)$
$F_c = \{A \rightarrow BC,\ E \rightarrow HA\}$

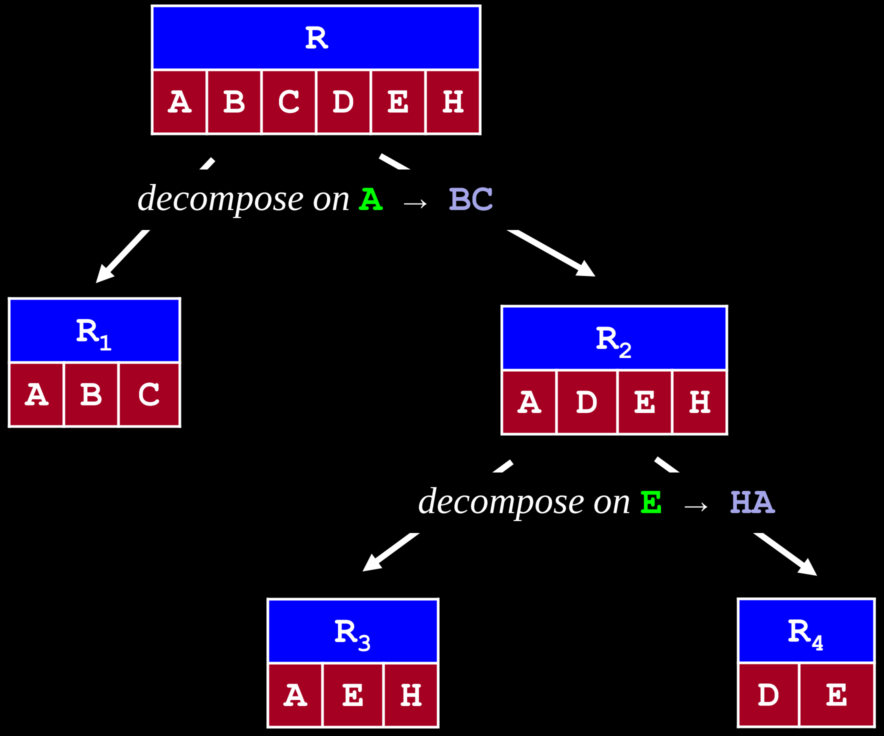
*Produce 2 BCNF decompositions: one that is DP and one that isn't*

# Boyce-Codd Normal Form (BCNF)

Example 2:

$R = (A, B, C, D, E, H)$
$F_c = \{A \rightarrow BC,\ E \rightarrow HA\}$

Decomposition #1:



# Boyce-Codd Normal Form (BCNF)

Example 2:

$R = (A, B, C, D, E, H)$
$F_c = \{A \rightarrow BC, E \rightarrow HA\}$

Decomposition #1:

$R = R_1 \cup R_3 \cup R_4$

| $R_1$ | | |
|---|---|---|
| A | B | C |

*covers* $A \rightarrow BC$

$\cup$

| $R_3$ | | |
|---|---|---|
| A | E | H |

*covers* $E \rightarrow HA$

$\cup$

| $R_4$ | |
|---|---|
| D | E |

*covers* –

$\Rightarrow$ *covers* $F_c$ $\Rightarrow$ *covers* $F^+$

*DP or not DP?*

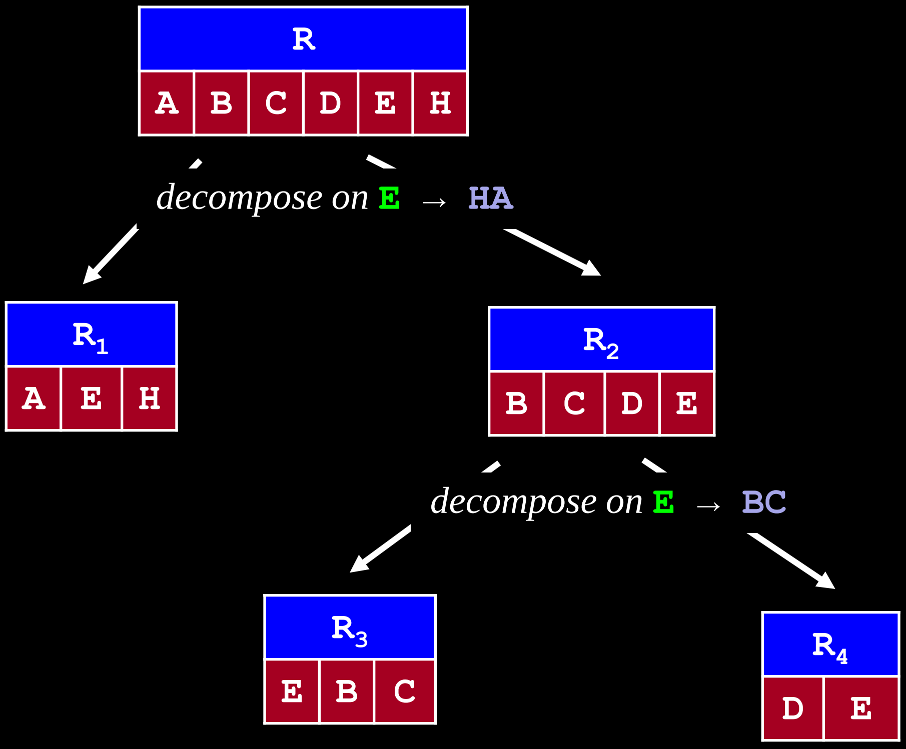
A: *DP*

# Boyce-Codd Normal Form (BCNF)

Example 2:

R = (A, B, C, D, E, H)
$F_c$ = {**A** → **BC**, **E** → **HA**}

Decomposition #2:



# Boyce-Codd Normal Form (BCNF)

Example 2:

$R = (A, B, C, D, E, H)$
$F_c = \{\mathbf{A} \rightarrow \mathbf{BC}, \mathbf{E} \rightarrow \mathbf{HA}\}$

Decomposition #2:

$R = R_1 \cup R_3 \cup R_4$

| $R_1$ | | |
|---|---|---|
| **A** | **E** | **H** |

*covers* **E** → **HA**

$\cup$

| $R_3$ | | |
|---|---|---|
| **E** | **B** | **C** |

*covers* **E** → **BC**

$\cup$

| $R_4$ | |
|---|---|
| **D** | **E** |

*covers* –

$\Rightarrow$ *does not cover* **A** → **B**

*DP or not DP?*

A: *Not DP*

# Boyce-Codd Normal Form (BCNF)

Example 3:

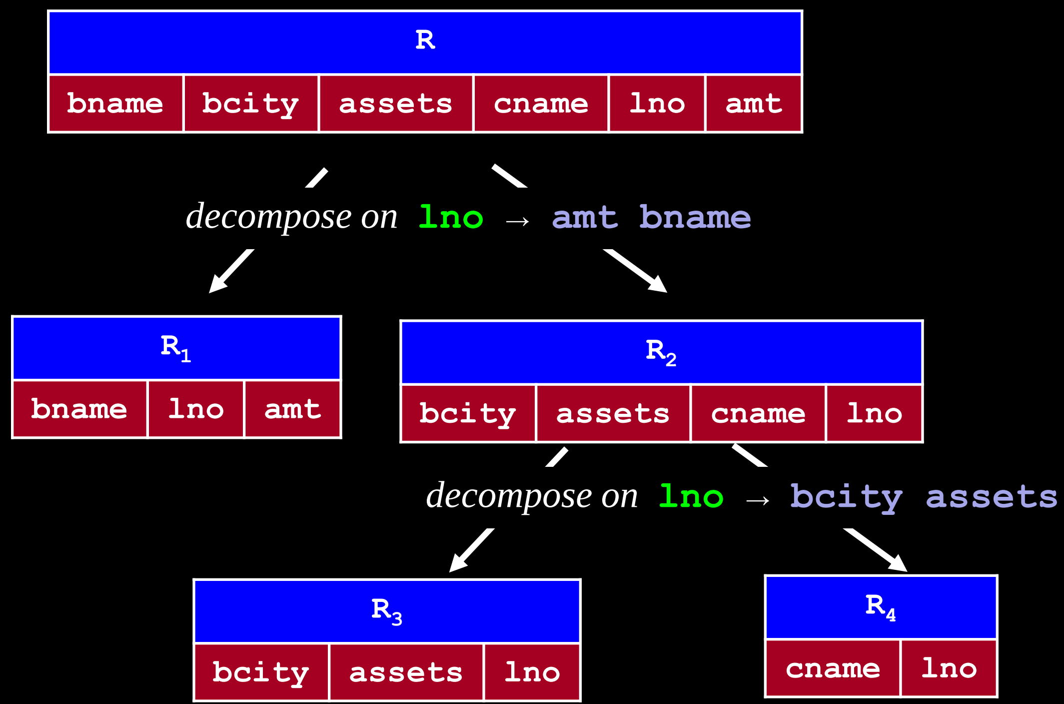
```
R  = (bname, bcity, assets, cname, lno, amt)
Fc = {lno   → amt bname,
      bname → bcity assets}
```

*Decompose* R *into BCNF, ensuring DP if possible*

# Boyce-Codd Normal Form (BCNF)

**Example 3:**

```
R  = (bname, bcity, assets, cname, lno, amt)
Fc = {lno   → amt bname,
      bname → bcity assets}
```



# Boyce-Codd Normal Form (BCNF)

**Example 3:**

R = (bname, bcity, assets, cname, lno, amt)
$F_c$ = {**lno** → **amt bname**,
**bname** → **bcity assets**}

**Result:**

$R = R_1 \cup R_3 \cup R_4$

| $R_1$ | | |
|---|---|---|
| bname | lno | amt |

*covers* **lno** → **amt bname**

$\cup$

| $R_3$ | | |
|---|---|---|
| lno | bcity | assets |

*covers* **lno** → **bcity assets**

$\cup$

| $R_4$ | |
|---|---|
| cname | lno |

*covers* –

⇒ *does not cover* **bname** → **bcity**

*DP or not DP?*

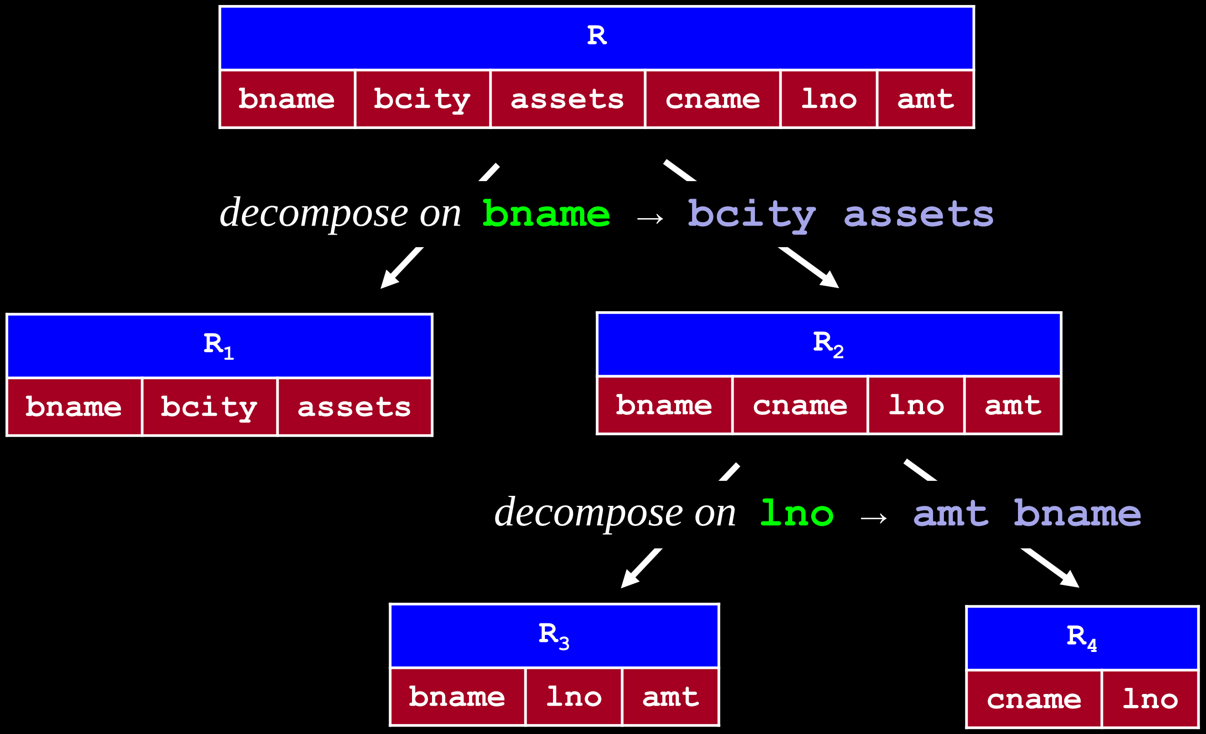
A: *Not DP*

# Boyce-Codd Normal Form (BCNF)

Example 3:

R = (bname, bcity, assets, cname, lno, amt)

$F_c$ = {**lno** → **amt bname**,
**bname** → **bcity assets**}



# Boyce-Codd Normal Form (BCNF)

**Example 3:**

$R$ = (bname, bcity, assets, cname, lno, amt)

$F_c$ = {**lno** → **amt bname**,
**bname** → **bcity assets**}

**Result:**

$$R = R_1 \cup R_3 \cup R_4$$

| R1 | | |
|---|---|---|
| bname | bcity | assets |

| R3 | | |
|---|---|---|
| bname | lno | amt |

| R4 | |
|---|---|
| cname | lno |

*covers* **bname** → **bcity assets**

$\cup$

*covers* **lno** → **amt bname**

$\cup$

*covers* –

$\Rightarrow$ *covers* $F_c$ $\Rightarrow$ *covers* $F^+$

*DP or not DP?*

A: *DP*

# Boyce-Codd Normal Form (BCNF)

## Ordering Decomposition Steps:

- Observe:

1) decompose on **`lno`** → **`amt bname`**
2) decompose on **`bname`** → **`bcity assets`**

*Not DP*

but

1) decompose on **`bname`** → **`bcity assets`**
2) decompose on **`lno`** → **`amt bname`**

*DP*

*Why?*

1) decompose on **`lno`** → **`amt bname`**

   $R_1 \leftarrow$ {lno, amt, bname}

   $R_2 \leftarrow$ R - {amt, bname} *(i.e.,* bname $\notin R_2$*)*

2) decompose on **`bname`** → **`bcity assets`**

*cannot apply to* $R_1$ *or* $R_2$*!!*

# Boyce-Codd Normal Form (BCNF)

## Ordering Decomposition Steps:

When applying FD's to decomposition, for any FD,

$$X \rightarrow \boxed{Y}$$

ensure that all FD's of the form,

$$\boxed{Y} \rightarrow Z$$

are applied before!

# Boyce-Codd Normal Form (BCNF)

**Example 4:**

R = (name, addr, fbeer, fmanf, lbeer, lmanf)

$F_c$ = {**name** → **addr fbeer**,
**lbeer** → **lmanf**,
**fbeer** → **fmanf**}

*Decompose* R *into BCNF, ensuring DP if possible*

**First…:**

R = (name, addr, fbeer, fmanf, lbeer, lmanf)

$F_c$ = {**name** → **addr $\boxed{\text{fbeer}}$**,
**lbeer** → **lmanf**,
**$\boxed{\text{fbeer}}$** → **fmanf**}

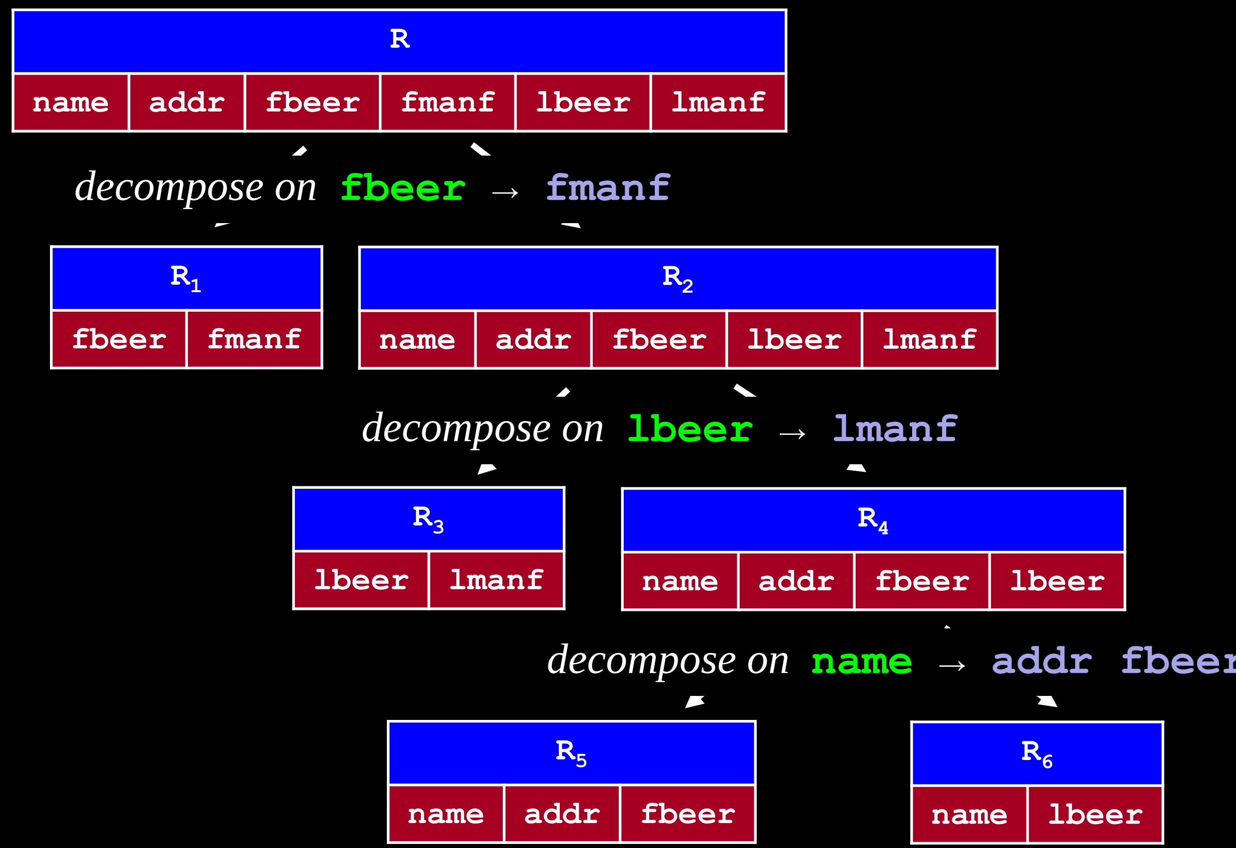
R = (name, addr, fbeer, fmanf, lbeer, lmanf)

$F_c$ = {**$\boxed{\text{fbeer}}$** → **fmanf**
**lbeer** → **lmanf**,
**name** → **addr $\boxed{\text{fbeer}}$**}

# Boyce-Codd Normal Form (BCNF)

**Example 4:**

```
R  = (name, addr, fbeer, fmanf, lbeer, lmanf)
Fc = { fbeer → fmanf
       lbeer → lmanf,
       name → addr fbeer}
```



# Boyce-Codd Normal Form (BCNF)

Example 4:

R = (name, addr, fbeer, fmanf, lbeer, lmanf)

$F_c$ = { **fbeer** → **fmanf**
**lbeer** → **lmanf**,
**name** → **addr fbeer**}

Result:

$R = R_1 \cup R_3 \cup R_5 \cup R_6$

| $R_1$ | |
|---|---|
| fbeer | fmanf |

*covers* **fbeer** → **fmanf**

∪

| $R_3$ | |
|---|---|
| lbeer | lmanf |

*covers* **lbeer** → **lmanf**

∪

| $R_5$ | | |
|---|---|---|
| name | addr | fbeer |

*covers* **name** → **addr fbeer**

∪

| $R_6$ | |
|---|---|
| name | lbeer |

*covers* –

$\Rightarrow$ *covers* $F_c$ $\Rightarrow$ *covers* $F^+$

*DP or not DP?*

A: *DP*

# Normalization

## BCNF

- can result in many decompositions (schemas) for same relation + FD Set
- all result schemas satisfy

    1. lossless joins
    2. redundancy avoidance
    3. dependency preservation (sometimes, but not always possible)

*When is it <u>impossible</u> for BCNF to satisfy DP?*

# Normalization

## BCNF

- can result in many decompositions (schemas) for same relation + FD Set
- all result schemas satisfy

1. lossless joins
2. redundancy avoidance
3. dependency preservation (sometimes, but not always possible)

*When is it <u>impossible</u> for BCNF to satisfy DP?*

A: *An example,*

```
R  = (J, K, L)
Fc = {L → K, JK → L}
```

*has no BCNF decomposition that is dependency preserving*

# Boyce-Codd Normal Form (BCNF)

Example 5:

$R = (J, K, L)$

$F_c = \{L \rightarrow K, JK \rightarrow L\}$

*Decompose* R *into BCNF*

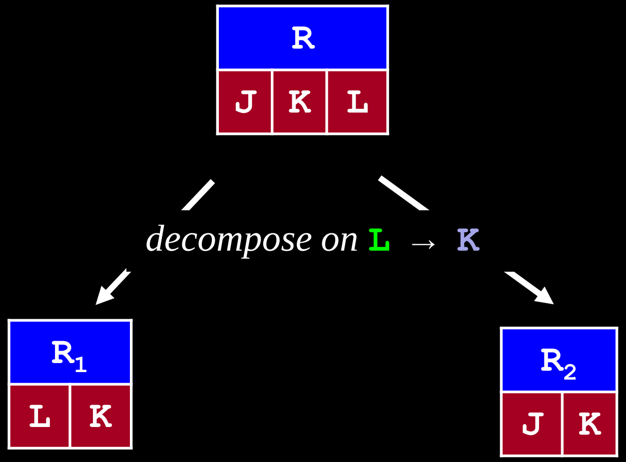
# Boyce-Codd Normal Form (BCNF)

Example 5:

$R = (J, K, L)$

$F_c = \{L \rightarrow K,\ JK \rightarrow L\}$

Decomposition #1



# Boyce-Codd Normal Form (BCNF)

Example 5:

$R = (J, K, L)$

$F_c = \{\mathbf{L} \rightarrow \mathbf{K},\ \mathbf{JK} \rightarrow \mathbf{L}\}$

Decomposition #1

$R = R_1 \cup R_2$

| $\mathbf{R_1}$ | |
|---|---|
| **L** | **K** |

*covers* **L** $\rightarrow$ **K**

$\cup$

| $\mathbf{R_2}$ | |
|---|---|
| **J** | **K** |

*covers nothing*

$\Rightarrow$ *does not cover* **JK** $\rightarrow$ **L**

*DP or not DP?*

A: *Not DP*

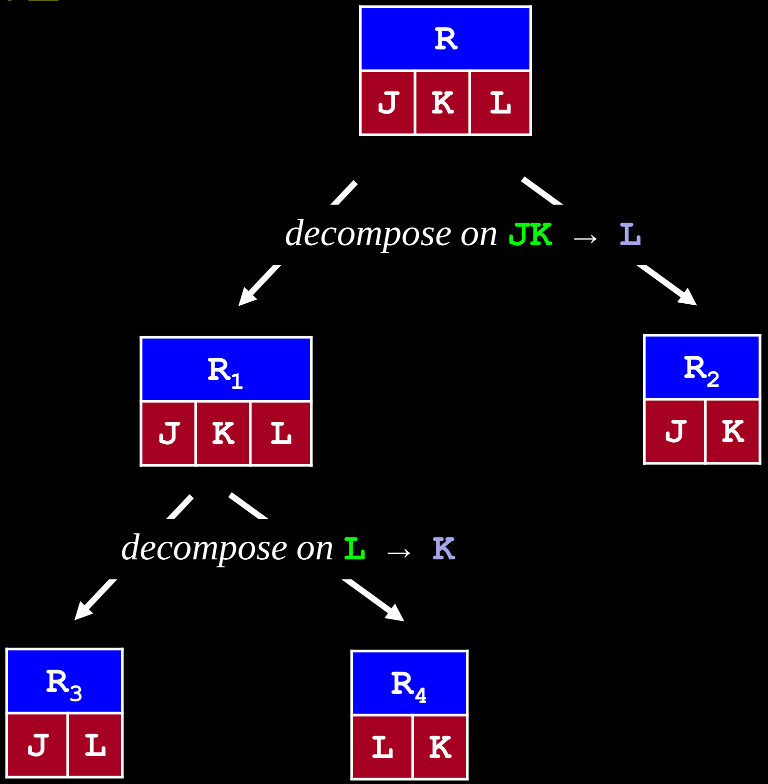
# Boyce-Codd Normal Form (BCNF)

Example 5:

R = (J, K, L)

$F_c$ = {**L** → **K**, **JK** → **L**}

Decomposition #2



# Boyce-Codd Normal Form (BCNF)

**Example 5:**

$R = (J, K, L)$

$F_c = \{L \rightarrow K,\ JK \rightarrow L\}$

**Decomposition #2**

$R = R_2 \cup R_3 \cup R_4$

| $R_2$ | |
|---|---|
| J | K |

*covers -*

$\cup$

| $R_3$ | |
|---|---|
| J | L |

*covers -*

$\cup$

| $R_4$ | |
|---|---|
| L | K |

*covers* $L \rightarrow K$

$\Rightarrow$ *does not cover* $JK \rightarrow L$

*DP or not DP?*

A: *Not DP*

# Boyce-Codd Normal Form (BCNF)

## Ordering Decomposition Steps:

When applying FD's to decomposition, for any FD,

$$X \rightarrow \boxed{Y}$$

ensure that all FD's of the form,

$$\boxed{Y} \rightarrow Z$$

are applied before!

## Not Always Possible

```
R  = (J, K, L)
Fc = {L → K, JK → L}
```

*which goes first?*

$\Rightarrow$ *DP not possible with BCNF*

# Boyce-Codd Normal Form (BCNF)

## Is This a Realistic Example?

```
R  = (J, K, L)
Fc = {L → K, JK → L}
```

A: *Yes*

```
R  = (CustName, BranchName, BankerName)
Fc = { BankerName → BranchName
       CustName BranchName → BankerName }
```

- *every banker works at one branch*
- *a customer works with the same banker at a given branch*

# Normalization Summary

## Theoretical Approach to DB Design based on FDs

- unlike ad hoc E/R approach, can know if designs are "good"
- good = satisfies some normal form

## Approach

- decompose universal relation in steps
- decomposition goals:
  1. Lossless Joins (LJ): preserve semantics
  2. Redundancy Avoidance (RA): no update anomalies
  3. Dependency Preservation (DP): fewer GICs to enforce

## Normal forms:

- BCNF: Guarantees LJ + RA
- 3NF: Guarantees LJ + DP (next class)