# COSI 127b
# Introduction to Database Systems

Lecture 16: Normalization (5)

# 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: 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)

# Review: 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: BCNF

## An Algorithm to Decompose A Relation into BCNF:

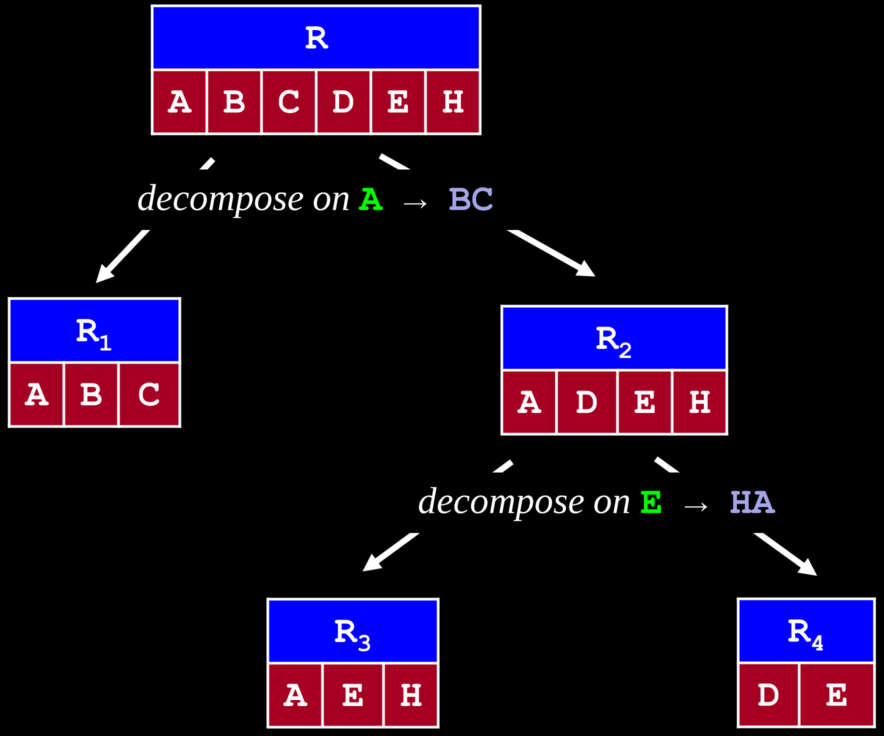
```
ALGORITHM BCNF (R: Relation, F: FD set)
 BEGIN
   Compute F+
   Result ← {R}
   WHILE some Ri ∈ Result not in BCNF DO
     Choose non-trivial (X → Y) ∈ F+ such that:
        •  X not a key of Ri, and
        •  (X → Y) covered by Ri
     Decompose Ri on (X → Y)
        •  Ri1 ← X ∪ Y
        •  Ri2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

END
```

# Review: BCNF

Example:

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

Decomposition:



# Review: BCNF

Example:

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

Decomposition:

$R = R_1 \cup R_3 \cup R_4$

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

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

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

$$\left.\begin{array}{l} \textit{covers } A \rightarrow BC \\ \cup \\ \textit{covers } E \rightarrow HA \\ \cup \\ \textit{covers } - \end{array}\right\} \Rightarrow \textit{covers } F_c \Rightarrow \textit{covers } F^+$$

*DP or not DP?*

A: *DP*

# Review: BCNF

Example:

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

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

First…:

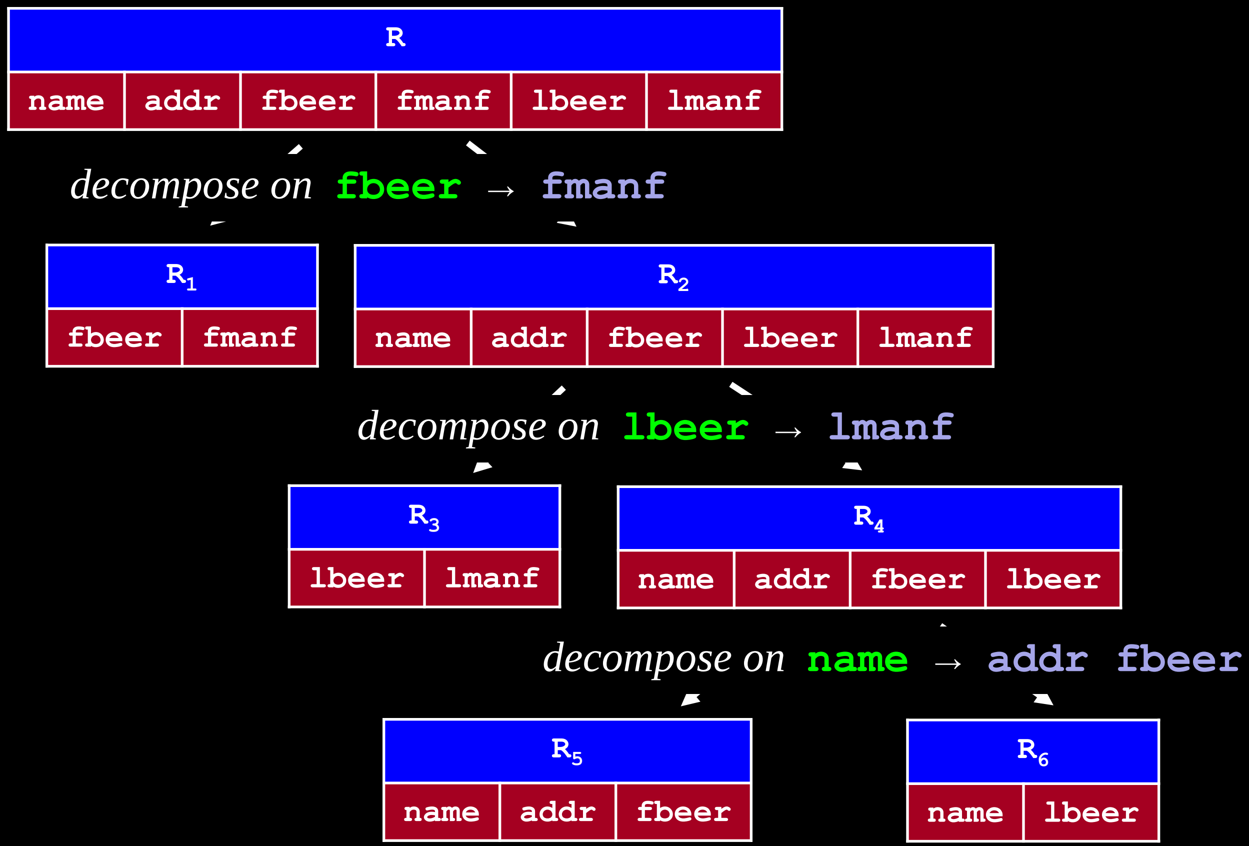
```
R  = (name, addr, fbeer, fmanf, lbeer, lmanf)
Fc = {name → addr [fbeer],
      lbeer → lmanf,
      [fbeer] → fmanf}
```

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

# Review: BCNF

Example:

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



# Review: BCNF

**Example:**

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

**Result:**

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

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

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

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

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

*covers* fbeer → fmanf

$\cup$

*covers* lbeer → lmanf

$\cup$

*covers* name → addr fbeer

$\cup$

*covers* –

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

*DP or not DP?*

A: *DP*

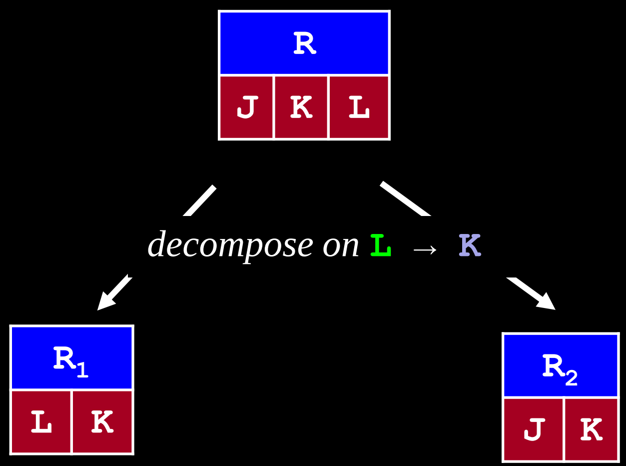
# Review: BCNF

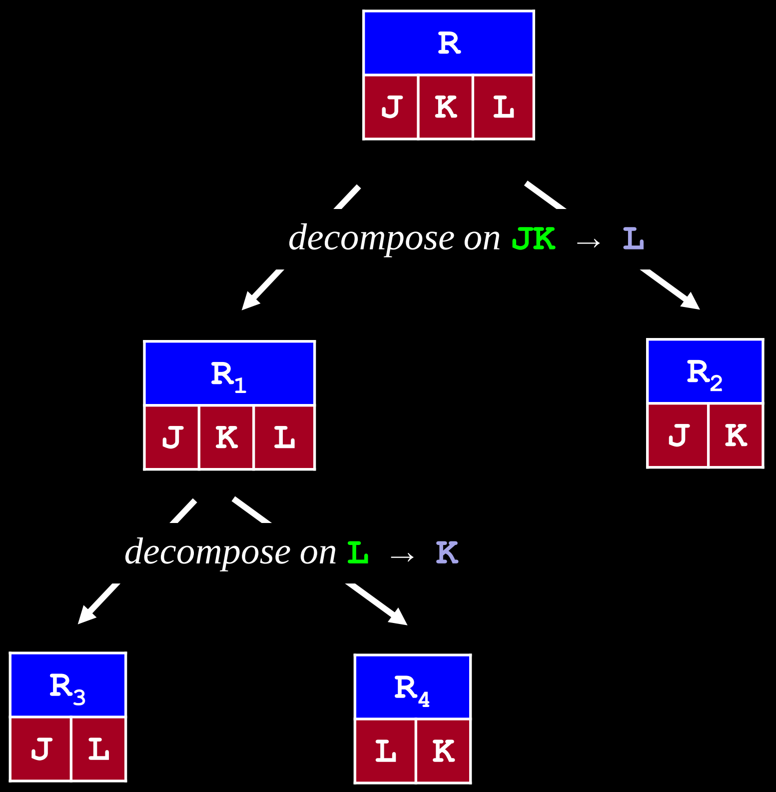
Example:

R = (J, K, L)

$F_c$ = {L → K, JK → L}

## No DP Decomposition



# 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)

# Third Normal Form (3NF)

## Motivation

- BCNF is not always dependency preserving
- for some apps, preserving dependencies more important than avoiding redundancy

## Solution: A weaker normal form (3NF)

- always exists a lossless-join, DP 3NF decomposition

# Third Normal Form (3NF)

## 3NF

Relation `R` in 3NF if for all $X \to Y$ in $F_c$, either of the following holds:

1. $X \to R$, or
2. each attribute in $Y - X$ is contained in a candidate key for `R`. (**NOTE**: each attribute may be in a different candidate key)

# Third Normal Form (3NF)

## Example:

R = (J, K, L)

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

*Candidate Keys:*

A: JK *and* JL

*In BCNF?*

A: *No -* **L** → **K** *covered but* L *not a key*

*BCNF + DP possible?*

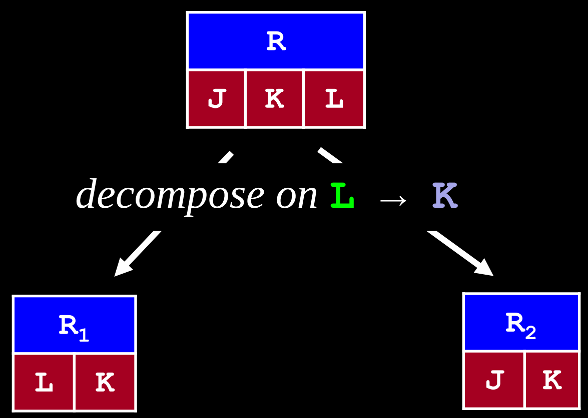
# Boyce-Codd Normal Form (BCNF)

## Example:

R = (J, K, L)

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

### BCNF Decomposition




$R_1 \cup R_2$ *does not cover* JK → L

# Third Normal Form (3NF)

## Example:

$R = (J, K, L)$
$F_c = \{JK \rightarrow L, L \rightarrow K\}$

*Candidate Keys?* A: JK *and* JL

*In BCNF?* A: *No* - $L \rightarrow K$ *covered but* L *not a key*

*BCNF + DP possible?* A: *No*

# Third Normal Form (3NF)

## Example:

$R = (J, K, L)$
$F_c = \{JK \rightarrow L, L \rightarrow K\}$

*Candidate Keys?* A: JK *and* JL

*In 3NF?* A: *Yes*

Relation R in 3NF if for all $X \rightarrow Y$ in $F_c$, either of the following holds:
- $X \rightarrow R$, or
- each attribute in $Y - X$ is contained in a candidate key for R

Consider the FD's in $F_c$:
- $JK \rightarrow L$ satisfies (1) ($JK \rightarrow R$)
- $L \rightarrow K$ satisfies (2) (K is contained in candidate key, JK)

# 3NF Decomposition

An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
```

# 3NF Decomposition

An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
```

# 3NF Decomposition

An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
     IF no R in Result includes X ∪ Y THEN
         • Construct R′ ← X ∪ Y
         • Result ← Result ∪ {R′}
```

# 3NF Decomposition

## An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
     IF no R in Result includes X ∪ Y THEN
         • Construct R′ ← X ∪ Y
         • Result ← Result ∪ {R′}
   IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
```

# 3NF Decomposition

## An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
     IF no R in Result includes X ∪ Y THEN
          •  Construct R′ ← X ∪ Y
          •  Result ← Result ∪ {R′}
   IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
   RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

## Candidate Keys?

1. {Custid, Empid}
2. {Custid, Bname}

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
         • Construct R′ ← X ∪ Y
         • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           •  Construct R′ ← X ∪ Y
           •  Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
        •  Construct R′ ← X (for any candidate key, X)
        •  Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           • Construct R′ ← X ∪ Y
           • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
         • Construct R′ ← X (for any candidate key, X)
         • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           • Construct R′ ← X ∪ Y
           • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
         • Construct R′ ← X (for any candidate key, X)
         • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           •  Construct R′ ← X ∪ Y
           •  Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
         •  Construct R′ ← X (for any candidate key, X)
         •  Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$, $R_3$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           • Construct R′ ← X ∪ Y
           • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
         • Construct R′ ← X (for any candidate key, X)
         • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$, $R_3$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$, $R_3$}

| $R_1$ | |
|---|---|
| Custid | Type |

| $R_2$ | |
|---|---|
| Empid | Bname |

| $R_3$ | | |
|---|---|---|
| Custid | Bname | Empid |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

$$R = R_1 \cup R_2 \cup R_3$$

| $R_1$ | |
|---|---|
| Custid | Type |

| $R_2$ | |
|---|---|
| Empid | Bname |

| $R_3$ | | |
|---|---|---|
| Custid | Bname | Empid |

## 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)

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

$R = R_1 \cup R_2 \cup R_3$

Empid → Bname *covered but* Empid *not a key!*

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

## 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)

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Empid → Bname**, **Custid, Bname → Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Custid, Bname → Empid**, **Empid → Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           • Construct R′ ← X ∪ Y
           • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Custid, Bname → Empid**, **Empid → Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           • Construct R′ ← X ∪ Y
           • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Custid, Bname → Empid**, **Empid → Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Custid, Bname** → **Empid**, **Empid** → **Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           •  Construct R′ ← X ∪ Y
           •  Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
         •  Construct R′ ← X (for any candidate key, X)
         •  Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Custid, Bname → Empid**, **Empid → Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
         •  Construct R′ ← X ∪ Y
         •  Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       •  Construct R′ ← X (for any candidate key, X)
       •  Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Same Example, Alternative FD Ordering:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid → Type**, **Custid, Bname → Empid**, **Empid → Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Custid, Bname** → **Empid**, **Empid** → **Bname**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

## Decomposition #1: $R = R_1 \cup R_2 \cup R_3$

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

*neither avoids redundancy, but #2 is preferable*

## Decomposition #2: $R = R_1 \cup R_2$

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

# 3NF Decomposition

## An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
     IF no R in Result includes X ∪ Y THEN
          •  Construct R′ ← X ∪ Y
          •  Result ← Result ∪ {R′}
   IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
   RETURN Result
END
```

# 3NF Decomposition

## An Algorithm to Decompose A Relation into 3NF (Revised):

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
     IF no R in Result includes X ∪ Y THEN
          •  Construct R′ ← X ∪ Y
          •  Result ← Result ∪ {R′}
   IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
   FOR each R, R' ∈ Result such that R ⊂ R' DO
     Result ← Result – {R}
   RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$, $R_3$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_2$ | |
|---|---|
| **Empid** | **Bname** |

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
          • Construct R′ ← X ∪ Y
          • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
    FOR each R, R' ∈ Result such that R ⊂ R' DO
       Result ← Result − {R}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_2$, $R_3$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| ~~$R_2$~~ | |
|---|---|
| ~~**Empid**~~ | ~~**Bname**~~ |

$\subset$

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           •  Construct R′ ← X ∪ Y
           •  Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
        •  Construct R′ ← X (for any candidate key, X)
        •  Result ← Result ∪ {R′}
    FOR each R, R' ∈ Result such that R ⊂ R' DO
       Result ← Result − {R}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

Result = {$R_1$, $R_3$}

| $R_1$ | |
|---|---|
| **Custid** | **Type** |

| $R_3$ | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
    Compute Fc
    Result := {}
    FOR each FD, (X → Y) ∈ Fc DO
      IF no R in Result includes X ∪ Y THEN
           • Construct R′ ← X ∪ Y
           • Result ← Result ∪ {R′}
    IF no R in Result includes a candidate key THEN
        • Construct R′ ← X (for any candidate key, X)
        • Result ← Result ∪ {R′}
    FOR each R, R' ∈ Result such that R ⊂ R' DO
       Result ← Result − {R}
    RETURN Result
END
```

# 3NF Decomposition

## Example:

R = (Custid, Empid, Bname, Type)

$F_c$ = {**Custid** → **Type**, **Empid** → **Bname**, **Custid, Bname** → **Empid**}

Candidate Keys: (1) {Custid, Empid} and (2) {Custid, Bname}

$$R = R_1 \cup R_3$$

| **$R_1$** | |
|---|---|
| **Custid** | **Type** |

| **$R_3$** | | |
|---|---|---|
| **Custid** | **Bname** | **Empid** |

*revised algorithm produces same decomposition regardless of FD order*

# Database 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

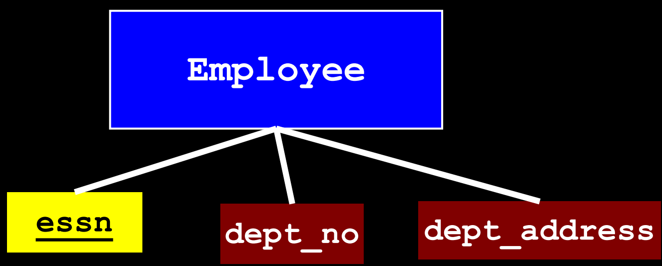
   Issue: Time-Consuming and Complex

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

# Ad hoc DB Design: E/R + Normalize

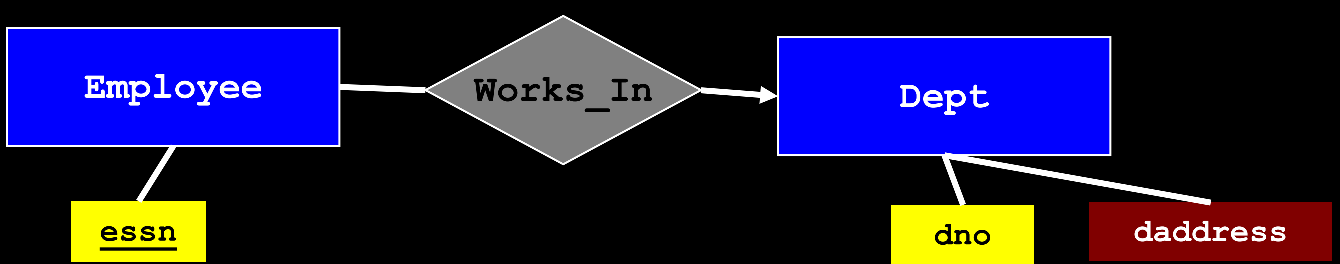
## Not all E/R Designs are correct…

- common E/R design flaw: functional dependencies from non-key attributes of an entity to other attributes of the entity




dept_no → dept_address

## Normalization detects bugs and informs how to fix



# Summary: Decomposition Goals

- Lossless Joins
- Redundancy Avoidance
- Dependency Preserving

# Summary: 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:

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

# Summary: 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)

# Summary: Normalization

## 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
```

# Summary: Normalization

## An Algorithm to Decompose A Relation into 3NF:

```
ALGORITHM 3NF (R: Relation, F: FD set)
 BEGIN
   Compute Fc
   Result := {}
   FOR each FD, (X → Y) ∈ Fc DO
     IF no R in Result includes X ∪ Y THEN
         •  Construct R′ ← X ∪ Y
         •  Result ← Result ∪ {R′}
   IF no R in Result includes a candidate key THEN
       • Construct R′ ← X (for any candidate key, X)
       • Result ← Result ∪ {R′}
   FOR each R, R' ∈ Result such that R ⊂ R' DO
     Result ← Result – {R}
   RETURN Result

END
```

# Course Evaluations

## Final Course Evaluations:

- end of term
- anonymous
- seen by all
- part of how instructor evaluated by university
- can suggest useful changes to course <u>next time it is offered</u>

## Midterm Course Evaluation:

- after midterm taken, before graded midterms returned
- anonymous
- seen by me only
- part of how I see if course is helping you
- can suggest useful changes to course <u>this semester</u>