COSI 127b
Introduction to Database Systems

Lecture 13: Normalization (3)

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 1st 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 and after midterm:

- DB Design using FDs and FD Algorithms

Review: Functional Dependencies

In General:

$A_1, \ldots, A_n \rightarrow B$

Informally:

If 2 tuples agree on their values for $A_1, \ldots, A_n$,
then they will also agree on their values for B

Formally:

$$\boxed{\forall t, u\ (t[A_1] = u[A_1] \land \ldots \land t[A_n] = u[A_n]) \Rightarrow t[B] = u[B])}$$

Review: FD Closures ($\mathtt{F}^+$)

Given FD sets over $\mathtt{R}$, $\mathtt{F}$ and $\mathtt{G}$, how to decide if $\mathtt{F} \equiv \mathtt{G}$?

- Idea: Compare sets of FDs that $\mathtt{F}$, $\mathtt{G}$ imply (closures)

$\mathtt{F} \equiv \mathtt{G}$ *if and only if* $\mathtt{F}^+ = \mathtt{G}^+$

Two ways to determine $\mathtt{F}^+$

- Attribute Closures
- Armstrong's Axioms

Review: FD Closures (F^+)

Algorithm 1: Using Attribute Closures

```
ALGORITHM FD-Closure (F: {FDs})
-- using Att-Closure
BEGIN
  Result ← {}
  Atts ← <all attributes appearing in FDs in F>
  FOREACH Z ⊆ Atts DO
   Result ← Result ∪ {Z → Att-Closure(Z,F)}
  RETURN Result
END
```

```
ALGORITHM Att-Closure (Z: {Attributes}, F: {FDs})
BEGIN
  Result ← Z
  REPEAT UNTIL STABLE
   FOR EACH functional dependency in F, X → Y DO
     IF X ⊆ Result THEN Result ← Result ∪ Y
  RETURN Result
END
```

Review: FD Closures (F^+)

Algorithm 2: Using Armstrong's Axioms

```
ALGORITHM FD-Closure (F: {FDs})
-- using Armstrong's Axioms
BEGIN
  Result ← F
  REPEAT UNTIL STABLE
    IF for any of Armstrong's Axioms (if A then B),
        A matches part of Result THEN
           Result ← Result ∪ B
  RETURN Result
END
```

Review: FD Closures (F^+)

Algorithm 2: Using Armstrong's Axioms

1. *Reflexivity*
   - if $Y \subseteq X$ then $X \to Y$
2. *Augmentation*
   - if $X \to Y$ then $WX \to WY$
3. *Transitivity*
   - if $X \to Y$ and $Y \to Z$ then $X \to Z$
4. *Union*
   - if $X \to Y$ and $X \to Z$ then $X \to YZ$
5. *Decomposition*
   - if $X \to YZ$ then $X \to Y$ and $X \to Z$
6. *Pseudotransitivity*
   - if $X \to Y$ and $WY \to Z$ then $WX \to Z$

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:
      $\{A \rightarrow B,\ B \rightarrow C\} < \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C\}$

   b) fewer attributes in FDs:
      $\{A \rightarrow B,\ B \rightarrow C\} < \{A \rightarrow BC,\ B \rightarrow C\}$

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
```

Normalization

Basic Idea:

1. Start with Universal Relation(s), `R`
    - all attributes in 1 table

Normalization

An Example Universal Relation:

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

Universal Relation: Expresses all "facts"

e.g.: Jones has a loan (L-17) for $1000 that was initiated at the Downtown branch in Brooklyn which has assets of $9M

Why Necessary to Decompose?

avoid unnecessary redundancy: update/deletion anomalies

Normalization

Basic Idea:

1. Start with Universal Relation(s), `R`
   - all attributes in 1 table

2. Determine FD set for `R` (`F`)

Review: Deriving FDs

FD Sources:

1. Key Constraints (e.g.: **`bname`** → **`Branch`**)
2. Known "many-to-one" (n::1) relationships
   - e.g.: **`beer`** → **`manufacturer`**, **`beer`** → **`price`**
3. Laws of Physics
   - e.g.: **`time, room`** → **`course`**
4. Trial-and-error
   - given `R =(A,B,C)`, see which of the following make sense:

A → B	A → C	B → A
B → C	C → A	C → B
AB → C	AC → B	BC → A

Normalization

Basic Idea:

1. Start with Universal Relation(s), `R`
   - all attributes in 1 table

2. Determine FD set for `R` (`F`)

3. Decompose `R` according to FDs in F^+

Decomposition

R					
bname	bcity	assets	cname	lno	amt

R_1			
bname	bcity	assets	cname

R_2		
cname	lno	amt

Notation for schema decomposition:

$R = R_1 \cup R_2$

BTW: Not a Good Decomposition

Decomposition

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

Decomposition

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	**bcity**	**assets**	**cname**
Downtown	Brooklyn	9M	Jones
Downtown	Brooklyn	9M	Jackson
Mianus	Horseneck	0.4M	Jones
Downtown	Brooklyn	9M	Williams

R_2		
cname	**lno**	**amt**
Jones	L-17	1000
Jackson	L-14	1500
Jones	L-93	500
Williams	L-17	1000

Decomposition

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	**bcity**	**assets**	**cname**
Downtown	Brooklyn	9M	Jones
Downtown	Brooklyn	9M	Jackson
Mianus	Horseneck	0.4M	Jones
Downtown	Brooklyn	9M	Williams

$\bowtie$

R_2		
cname	**lno**	**amt**
Jones	L-17	1000
Jackson	L-14	1500
Jones	L-93	500
Williams	L-17	1000

=

bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jones	L-93	500
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	1.7M	Jones	L-17	1000
Mianus	Horseneck	1.7M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

Lossy join: by adding noise, have lost meaningful info as a result of decomposition

Decomposition

R					
bname	bcity	assets	cname	lno	amt

R_1			
bname	bcity	assets	cname

R_2		
cname	lno	amt

$R = R_1 \cup R_2$?

BTW: Not a Good Decomposition

Goals of Decomposition

1. Lossless Joins
   - must be able to reconstruct universal relation via natural join of tables resulting from decomposition

2. Redundancy Avoidance
   - want to avoid unnecessary data duplication

3. Dependency Preservation
   - want to minimize the cost of global integrity constraints based on functional dependencies (i.e.: avoid big joins in assertions)

Goals of Decomposition

1. Lossless Joins
   - Avoid information loss

2. Redundancy Avoidance
   - Avoid update anomalies

3. Dependency Preservation
   - Avoid expensive global integrity constraints

Relative Importance:

1: Primary Importance

2,3: Secondary Importance

Goal #1: Lossless Joins

Intuition: a bad decomposition revisited

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	**bcity**	**assets**	**cname**
Downtown	Brooklyn	9M	Jones
Downtown	Brooklyn	9M	Jackson
Mianus	Horseneck	0.4M	Jones
Downtown	Brooklyn	9M	Williams

R_2		
cname	**lno**	**amt**
Jones	L-17	1000
Jackson	L-14	1500
Jones	L-93	500
Williams	L-17	1000

Goal #1: Lossless Joins

Intuition: a bad decomposition revisited

R					
bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	bcity	assets	cname
Downtown	Brooklyn	9M	Jones
Downtown	Brooklyn	9M	Jackson
Mianus	Horseneck	0.4M	Jones
Downtown	Brooklyn	9M	Williams

$\bowtie$

R_2		
cname	lno	amt
Jones	L-17	1000
Jackson	L-14	1500
Jones	L-93	500
Williams	L-17	1000

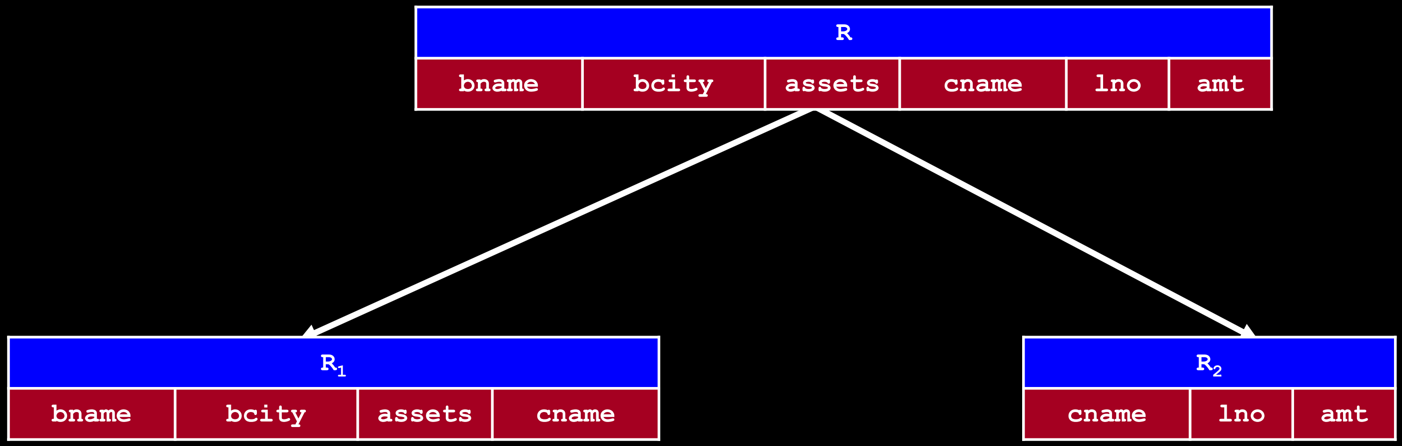
=

bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jones	L-93	500
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	1.7M	Jones	L-17	1000
Mianus	Horseneck	1.7M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

A: *lossy*

Goal #1: Lossless Joins

Intuition: a bad decomposition revisited



$R = R_1 \cup R_2$ *is lossy*

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	**bcity**	**assets**	**cname**
Downtown	Brooklyn	9M	Jones
Downtown	Brooklyn	9M	Jackson
Mianus	Horseneck	0.4M	Jones
Downtown	Brooklyn	9M	Williams

R_2		
bname	**lno**	**amt**
Downtown	L-17	1000
Downtown	L-14	1500
Mianus	L-93	500

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?

R					
bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	bcity	assets	cname
Downtown	Brooklyn	9M	Jones
Downtown	Brooklyn	9M	Jackson
Mianus	Horseneck	0.4M	Jones
Downtown	Brooklyn	9M	Williams

$\bowtie$

R_2		
bname	lno	amt
Downtown	L-17	1000
Downtown	L-14	1500
Mianus	L-93	500

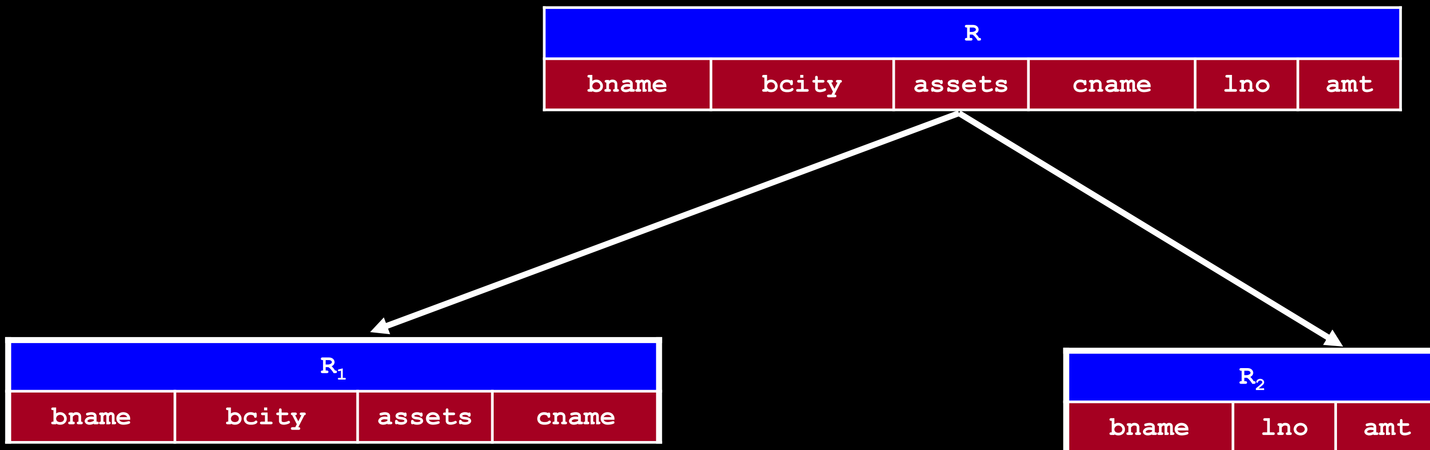
=

bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jones	L-14	1500
Downtown	Brooklyn	9M	Jackson	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000
Downtown	Brooklyn	9M	Williams	L-14	1500

A: *lossy*

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?



$R = R_1 \cup R_2$ *is lossy*

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	**assets**	**cname**	**lno**
Downtown	9M	Jones	L-17
Downtown	9M	Jackson	L-14
Mianus	0.4M	Jones	L-93
Downtown	9M	Williams	L-17

R_2		
lno	**bcity**	**amt**
L-17	Brooklyn	1000
L-14	Brooklyn	1500
L-93	Horseneck	500

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1			
bname	**assets**	**cname**	**lno**
Downtown	9M	Jones	L-17
Downtown	9M	Jackson	L-14
Mianus	0.4M	Jones	L-93
Downtown	9M	Williams	L-17

$\bowtie$

R_2		
lno	**bcity**	**amt**
L-17	Brooklyn	1000
L-14	Brooklyn	1500
L-93	Horseneck	500

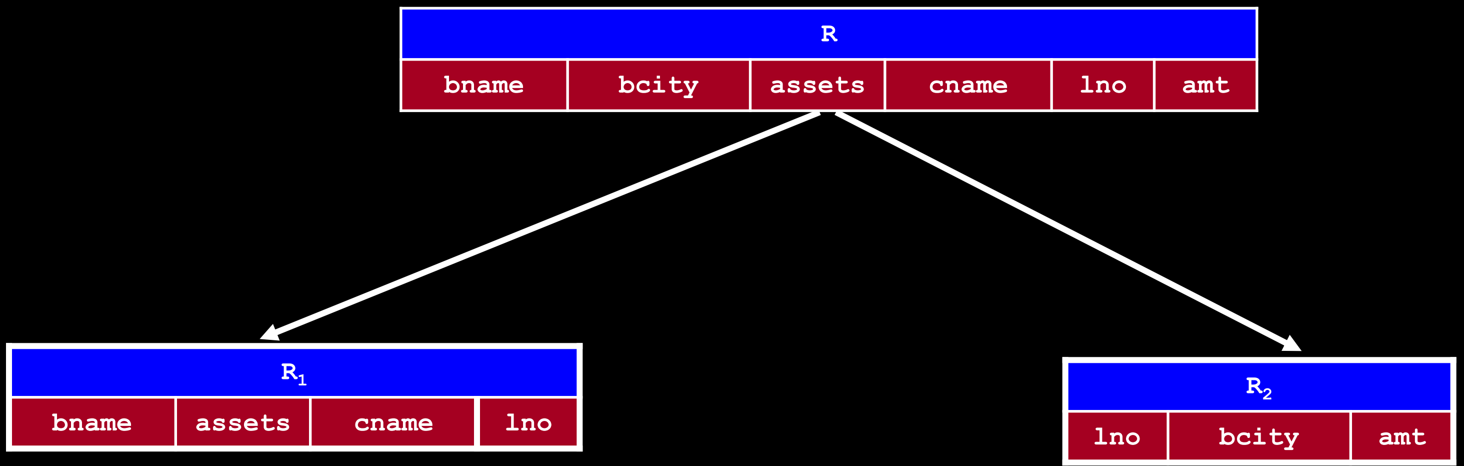
=

bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

A: *lossless*

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?



$R = R_1 \cup R_2$ *is lossless*

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?

R					
bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1		
bname	bcity	assets
Downtown	Brooklyn	9M
Mianus	Horseneck	0.4M

R_2			
bname	cname	lno	amt
Downtown	Jones	L-17	1000
Downtown	Jackson	L-14	1500
Mianus	Jones	L-93	500
Downtown	Williams	L-17	1000

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?

R					
bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

R_1		
bname	bcity	assets
Downtown	Brooklyn	9M
Mianus	Horseneck	0.4M

$\bowtie$

R_2			
bname	cname	lno	amt
Downtown	Jones	L-17	1000
Downtown	Jackson	L-14	1500
Mianus	Jones	L-93	500
Downtown	Williams	L-17	1000

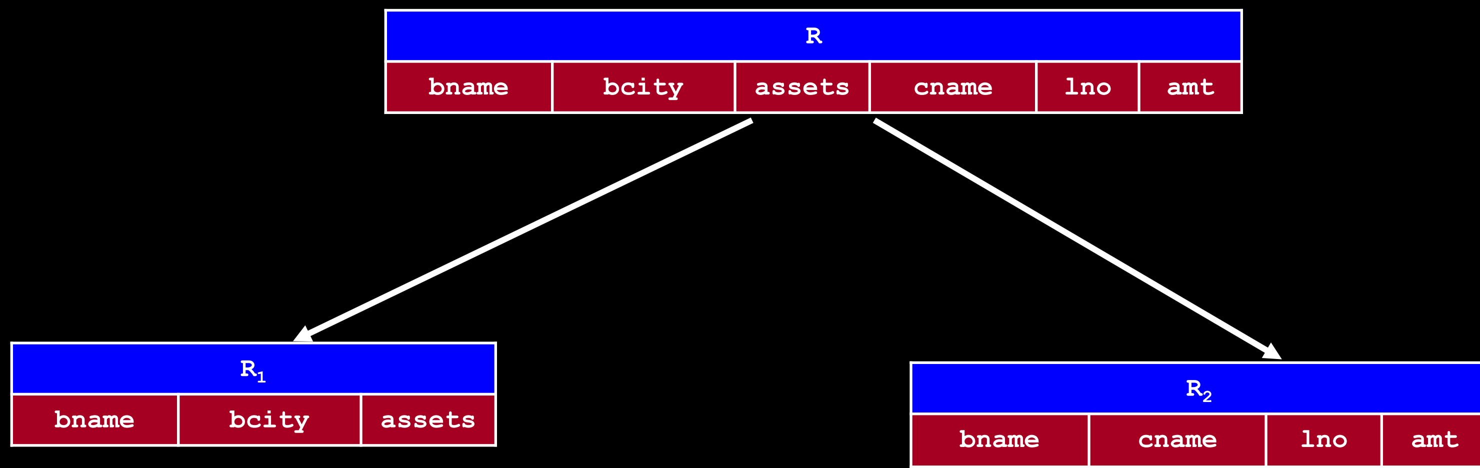
=

A: *lossless*

bname	bcity	assets	cname	lno	amt
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

Goal #1: Lossless Joins

Intuition: is decomposition lossless or lossy?



$R = R_1 \cup R_2$ *is lossless*

When is decomposition lossless?

Goal #1: Lossless Joins

Test: decomposition step $R = R_1 \cup R_2$ lossless iff

$$R_1 \cap R_2 \rightarrow R_1, \text{ or}$$
$$R_1 \cap R_2 \rightarrow R_2$$

- i.e., intersecting atts must be superkey for either result relation

Example: $|\mathbf{R}| = 4$

R					
bname	**bcity**	**assets**	**cname**	**lno**	**amt**
Downtown	Brooklyn	9M	Jones	L-17	1000
Downtown	Brooklyn	9M	Jackson	L-14	1500
Mianus	Horseneck	0.4M	Jones	L-93	500
Downtown	Brooklyn	9M	Williams	L-17	1000

Goal #1: Lossless Joins

Test: decomposition step $\mathtt{R} = \mathtt{R}_1 \cup \mathtt{R}_2$ <u>lossless</u> iff

$$\boxed{\begin{array}{l} \mathbf{R_1} \cap \mathbf{R_2} \rightarrow \mathbf{R_1}, \text{ or} \\ \mathbf{R_1} \cap \mathbf{R_2} \rightarrow \mathbf{R_2} \end{array}}$$

- i.e., intersecting atts must be superkey for either result relation

Example: $|\mathbf{R}| = 4$

R_1			
bname	**assets**	**cname**	**lno**
Downtown	9M	Jones	L-17
Downtown	9M	Jackson	L-14
Mianus	0.4M	Jones	L-93
Downtown	9M	Williams	L-17

R_2		
lno	**bcity**	**amt**
L-17	Brooklyn	1000
L-14	Brooklyn	1500
L-93	Horseneck	500

- `lno` a key $\Rightarrow$ n::1 relationship

- `lno` not a key $\Rightarrow |\mathtt{R}_1| = 4$

$\therefore$ 4 tuples in $\mathtt{R}_1 \bowtie \mathtt{R}_2$

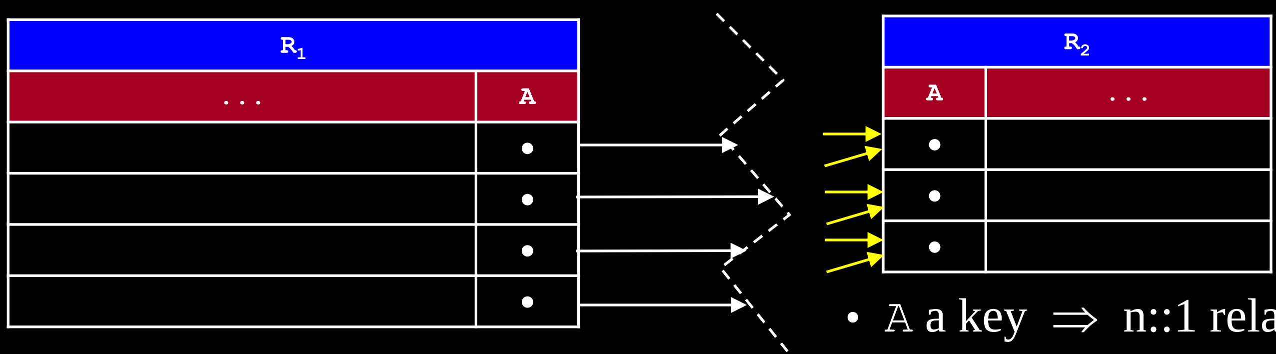
Goal #1: Lossless Joins

Test: decomposition step $R = R_1 \cup R_2$ <u>lossless</u> iff

$$\boxed{\begin{array}{l} R_1 \cap R_2 \rightarrow R_1, \text{ or} \\ R_1 \cap R_2 \rightarrow R_2 \end{array}}$$

- i.e., intersecting atts must be superkey for either result relation

In General: Suppose **R** has n tuples and $\mathbf{R_1} \cap \mathbf{R_2} = \mathbf{A}$



- A not a key $\Rightarrow |R_1| = n$
- A a key $\Rightarrow$ n::1 relationship

$\therefore$ n tuples in $R_1 \bowtie R_2$

Tests for Decomposition Goals

Test of $\mathtt{R} = \mathtt{R}_1 \cup \ldots \cup \mathtt{R}_n$ with FD set $\mathtt{F}$:

- Lossless Joins? iff for each decomposition step, $\mathtt{R}_k = \mathtt{R}_i \cup \mathtt{R}_j$:

  common atts of result relations form a key for one of them

Tests for Decomposition Goals

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

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

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

+

Test for Lossless Joins

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

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

Is the decomposition of R lossless?

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

A → **R_1**
B → **R_2**

2) What is $R_1 \cap R_2$?

B

3) Does **$R_1 \cap R_2$** → **R_1** or **$R_1 \cap R_2$** → **R_2**

Yes, **B** → **R_2**

Therefore, decomposition of R is <u>lossless</u>

Test for Lossless Joins

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

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

Is the decomposition of R lossless?

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

$A \rightarrow R_1$
$B \rightarrow R_2$

2) What is $R_1 \cap R_2$?

C

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

No.

Therefore, decomposition of R is <u>lossy.</u>

Test for Lossless Joins

Example	Lossless Joins?
R_1 = (A,B) R_2 = (B,C) F = {A → B, B → C}	yes
R_1 = (A,C) R_2 = (B,C) F = {A → B, B → C}	no

Test for Lossless Joins

Example 3: $R = R_1 \cup R_2$ $(R = (A,B,C))$

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

Is the decomposition of R lossless?

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

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

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, $C \rightarrow R_2$

Therefore, decomposition of R is lossless

Test for Lossless Joins

Example 4: $R = R_1 \cup R_2$ $(R = (A,B,C))$

$R_1 = (A,B,C)$

$R_2 = (B,C)$

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

Is the decomposition of R lossless?

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

$AB \rightarrow R_1$

$C \rightarrow R_2$

Any Others?

2) What is $R_1 \cap R_2$?

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

Test for Lossless Joins

Example 4: $R = R_1 \cup R_2$ $(R = (A,B,C))$

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

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 <u>lossless</u>

Test for Lossless Joins

Example	Lossless Joins?
$R_1 = (A,B)$ $R_2 = (B,C)$ $F = \{A \rightarrow B,\ B \rightarrow C\}$	yes

Test for Lossless Joins

Example	Lossless Joins?
R_1 = (A,B) R_2 = (B,C) F = {**A** → **B**, **B** → **C**}	yes
R_1 = (A,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes
R_1 = (A,B,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes

Goal #2: Redundancy Avoidance

Intuition: when is there redundancy in a relation, R?

R		
A	**B**	**C**
a	x	1
e	x	1
g	y	2
h	y	2
m	z	1

Which att(s) in R show redundancy? A: B, C

What apparent FD involves these atts? A: **B** $\rightarrow$ **C**

What is the only candidate key of R? A: A

In general, when is there redundancy in a relation, R?

A: When an FD in F^+, **X** $\rightarrow$ **Y** is "covered" by R
(i.e., X, Y $\in$ R) but **X** $\nrightarrow$ **R**

Tests for Decomposition Goals

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:

  all nontrivial FDs covered by R_i have key for lhs

Tests for Decomposition Goals

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, $\mathbf{X} \rightarrow \mathbf{R_i}$

+

Test for Redundancy

Single Table Example: $R = (A,B,C)$

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

Does R have redundancy?

A: 1) What are the candidate keys of R?

$A \rightarrow R$

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

$A \rightarrow BC,\ A \rightarrow B,\ A \rightarrow C$ ($A \rightarrow R$) ✓

$B \rightarrow C$ ($B \not\rightarrow R$ because $B \not\rightarrow A$) ✗

Therefore, R <u>has</u> redundancy

Test for Redundancy

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

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

Does decomposition of R have redundancy?

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

$A \rightarrow R_1$
$B \rightarrow R_2$

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

$A \rightarrow B$ ($A \rightarrow R_1$) ✓

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

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

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

Test for Redundancy

Example 3: $R = R_1 \cup R_2$ ($R = (A,B,C)$)

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

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

Test for Redundancy

Example 4: $R = R_1 \cup R_2$ $(R = (A,B,C))$

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

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 <u>has</u> redundancy

Test for Redundancy

Example	Lossless Joins?
R_1 = (A,B) R_2 = (B,C) F = {**A** → **B**, **B** → **C**}	yes
R_1 = (A,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes
R_1 = (A,B,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes

Test for Redundancy

Example	Lossless Joins?	Avoids Redundancy?
R_1 = (A,B) R_2 = (B,C) F = {**A** → **B**, **B** → **C**}	yes	yes
R_1 = (A,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes	yes
R_1 = (A,B,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes	no

Goal #3: Dependency Preservation

Intuition: enforcing functional dependencies

Consider:

R	
A	**B**

To enforce the FD, **A** → **B** over R:

A → **B**

```
CREATE ASSERTION A-B
  CHECK (NOT EXISTS
    (SELECT *
     FROM R AS r1, R AS r2
     WHERE r1.A = r2.A AND r1.B <> r2.B))
```

Goal #3: Dependency Preservation

Intuition: enforcing functional dependencies

Consider:

R	
A	B

S	
A	C

To enforce the FDs, **A** → **B** , **A** → **C** over R, S:

$A \rightarrow B$

```
CREATE ASSERTION A-B
  CHECK (NOT EXISTS
    (SELECT *
     FROM R AS r1, R AS r2
     WHERE r1.A = r2.A AND r1.B <> r2.B))
```

+

$A \rightarrow C$

```
CREATE ASSERTION A-C
  CHECK (NOT EXISTS
    (SELECT *
     FROM S AS s1, S AS s2
     WHERE s1.A = s2.A AND s1.C <> s2.C))
```

Goal #3: Dependency Preservation

Intuition: enforcing functional dependencies

Consider:

R		
A	B	C

To enforce the FDs, **A** → **B** , **A** → **C** over R:

A:

A → **BC**

```
CREATE ASSERTION A-BC
  CHECK (NOT EXISTS
    (SELECT *
     FROM R AS r1, R AS r2
     WHERE r1.A = r2.A AND ((r1.B <> r2.B) OR (r1.C <> r2.C))
```

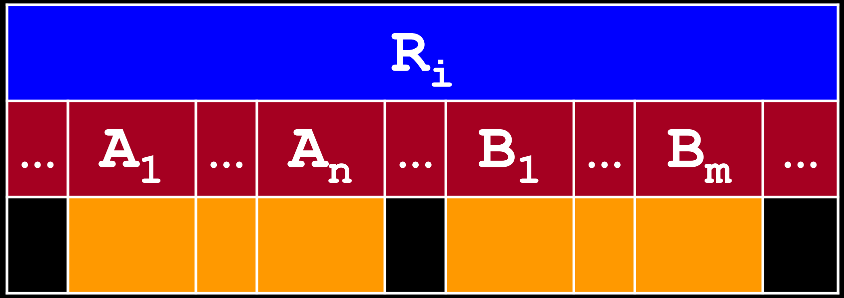
Idea: **X** → **Y** less $ to enforce if X, Y in same table

Goal #3: Dependency Preservation

Intuition: enforcing functional dependencies

ensure 1 table examined per FD
(table appears 2x in assertion `FROM` clause)

i.e.: decomposed tables should still cover FDs



e.g., R_i covers $A_1 \ldots A_n \rightarrow B_1 \ldots B_m$

Test for Dependency Preservation

Is $R = R_1 \cup \ldots \cup R_n$ with FD's, F, dependency preserving?

Test:

1. Compute F^+

2. Compute G: FD's in F^+ covered by individual tables, $R_1, \ldots, R_n$

```
G ← ∅
FOR i ← 1 TO n DO
    Add to G those FD's in F+ that are covered by Ri
```

3. Test if $F^+ = G^+$

   - if yes, decomposition is dependency preserving
   - if no, decomposition is not dependency preserving
     → $(F^+ - G^+)$ not covered by the decomposition

Test for Dependency Preservation

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

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

Is decomposition of R dependency preserving?

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

$A \rightarrow B$

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

$B \rightarrow C$

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

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

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

Test for Dependency Preservation

Example 3: $R = R_1 \cup R_2$ ($R = (A,B,C)$)

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

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 <u>is not</u> dependency preserving

Test for Dependency Preservation

Example 4: $R = R_1 \cup R_2$ $(R = (A,B,C))$

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

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

Summary: Examples

Example	Lossless Joins?	Avoids Redundancy?
$R_1 = (A,B)$ $R_2 = (B,C)$ $F = \{A \rightarrow B,\ B \rightarrow C\}$	yes	yes
$R_1 = (A,C)$ $R_2 = (B,C)$ $F = \{AB \rightarrow C,\ C \rightarrow B\}$	yes	yes
$R_1 = (A,B,C)$ $R_2 = (B,C)$ $F = \{AB \rightarrow C,\ C \rightarrow B\}$	yes	no

Summary: Examples

Example	Lossless Joins?	Avoids Redundancy?	Dependency Preserving?
R_1 = (A,B) R_2 = (B,C) F = {**A** → **B**, **B** → **C**}	yes	yes	yes
R_1 = (A,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes	yes	no
R_1 = (A,B,C) R_2 = (B,C) F = {**AB** → **C**, **C** → **B**}	yes	no	yes

Tests for Decomposition Goals

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, $\mathbf{X} \rightarrow \mathbf{R_i}$

- Dependency Preserving? iff:

  FDs covered by single relations R_i are equivalent to F

Tests for Decomposition Goals

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: Goals of Decomposition

Goal	Motivation
Lossless Joins	avoid info loss
Redundancy Avoidance	avoid update and deletion anomalies
Dependency Preservation	efficient FD enforcement

Summary: Goals of Decomposition

Goal	Motivation	Idea
Lossless Joins	avoid info loss	recomposing tables should not add noise
Redundancy Avoidance	avoid update and deletion anomalies	only FD’s with keys covered by decomposed tables
Dependency Preservation	efficient FD enforcement	fewer global ICs required to enforce FDs

Summary: Goals of Decomposition

Goal	Motivation	Idea	Test
Lossless Joins	avoid info loss	recomposing tables should not add noise	For: $R = R_1 \cup R_2$ $(R_1 \cap R_2) \rightarrow R_1$ or $(R_1 \cap R_2) \rightarrow R_2$
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: $R = R_1 \cup \ldots \cup R_n$ (FD's covered by each R_i)$^+$ = (FD's covered by R)$^+$

Summary: Goals of Decomposition

Goal	Motivation	Idea	Test	Guaranteed By
Lossless Joins	avoid info loss	recomposing tables should not add noise	For: $R = R_1 \cup R_2$ $(R_1 \cap R_2) \rightarrow R_1$ or $(R_1 \cap R_2) \rightarrow R_2$	**BCNF, 3NF**
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	**BCNF**
Dependency Preservation	efficient FD enforcement	fewer global ICs required to enforce FDs	For: $R = R_1 \cup \ldots \cup R_n$ (FD's covered by each R_i)$^+$ = (FD's covered by R)$^+$	**3NF**

After the midterm: 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

Midterm Coverage

Topic	Lecture(s)	Assignment(s)
Introduction	1	-
Relational Data Model	2	-
Relational Algebra	2, 3	PS 1
Relational Calculus	3, 4	PS 1, PS 2
SQL	5, 6	PS 1, PA 1
Transactions	7	PS 3
Integrity Constraints	8, 9	PS 3
E/R Data Model	9, 10, 11	PS 4
Functional Dependencies	9, 11	PS 4

Summary: Topics on the Midterm

1. Data Organization
   - Logical: Relational Data Model, Database Design

2. Data Retrieval
   - Logical: Query Languages: RA, TRC, SQL

3. Data Integrity
   - Logical: Transactions, Integrity Constraints

Summary: Topics on the Midterm

1. Data Organization

Includes:

- Relational Terminology
- E/R Data Model
- E/R $\rightarrow$ Relation Xlation
- Functional Dependencies

Does not include:

- Canonical Covers of FDs
- Decomposition

2. Data Retrieval

- Logical: Query Languages: RA, TRC, SQL

3. Data Integrity

- Logical: Transactions, Integrity Constraints

Summary: Topics on the Midterm

1. Data Organization

Includes:

- Relational Terminology
- E/R Data Model
- E/R → Relation Xlation
- Functional Dependencies

Does not include:

- Canonical Covers of FDs
- Decomposition

2. Data Retrieval

Includes:

- Relational Algebra
- Tuple Relational Calculus
- SQL (DML, DDL, Views)
- Xlations: RA ↔ TRC ↔ SQL

3. Data Integrity

- Logical: Transactions, Integrity Constraints

Summary: Topics on the Midterm

1. Data Organization

Includes:

- Relational Terminology
- E/R Data Model
- E/R → Relation Xlation
- Functional Dependencies

Does not include:

- Canonical Covers of FDs
- Decomposition

2. Data Retrieval

Includes:

- Relational Algebra
- Tuple Relational Calculus
- SQL (DML, DDL, Views)
- Xlations: RA ↔ TRC ↔ SQL

3. Data Integrity

Includes:

- ACID Properties
- Serializability and Conflict Serializability
- Alternative SQL Isolation Policies and Isolation Anomalies
- Integrity Constraints
- GICs and FDs
- FD Closures (Attribute Closures, Armstrongs Axioms)

Midterm

Studying Suggestions

- Review slides and text
- Review homework solutions (make sure you understand)
- Practice Exercises in text (solutions in back)
- Study groups
- Set high bar for "understanding"