CMPT 354: Database System I

Lecture 9. Design Theory

Design Theory

- **<u>Design theory</u>** is about how to represent your data to avoid anomalies.

Design 1

Student	Course	Room
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

Design 2

Student	Course
Mike	354
Mary	354
Sam	354
..	..

Course	Room
354	AQ3149
454	T9204

Four Types of Anomalies - 1

- What's wrong?

Student	Course	Room
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

If every course is in only one room, contains ***redundant*** information!

Four Types of Anomalies - 2

- What's wrong?

Student	Course	Room
Mike	354	AQ3149
Mary	354	T9204
Sam	354	AQ3149
..	..	..

If we update the room number for one tuple, we get inconsistent data = an ***update* anomaly**

Four Types of Anomalies - 3

- What's wrong?

Student	Course	Room
..	..	..

If everyone drops the class, we lose what room the class is in! = a ***delete* anomaly**

Four Types of Anomalies - 4

- What’s wrong?

...	454	T9204

Student	Course	Room
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

Similarly, we can’t reserve a room without students = an ***insert* anomaly**

Elimination of Anomalies

- Is it better?

Student	Course
Mike	354
Mary	354
Sam	354
..	..

Course	Room
354	AQ3149
454	T9204

- Redundancy?
- Update anomaly?
- Delete anomaly?
- Insert anomaly?

How to find this *decomposition?*

Normal Forms

- 1st Normal Form (1NF) = All tables are flat
- *2nd Normal Form = disused*
- Boyce-Codd Normal Form (BCNF) = no bad FDs
- 3rd , 4th , and 5th Normal Forms = see text books

1^{st} Normal Form (1NF)

Student	Courses
Mary	{CS145,CS229}
Joe	{CS145,CS106}
…	…

Violates 1NF.

Student	Courses
Mary	CS145
Mary	CS229
Joe	CS145
Joe	CS106

In 1^{st} NF

1NF Constraint: Types must be atomic!

Normal Forms

- 1st Normal Form (1NF) = All tables are flat
- *2nd Normal Form = disused*
- **Boyce-Codd Normal Form (BCNF) = no bad FDs**
- 3rd , 4th , and 5th Normal Forms = see text books

What’s this?

Outline

1. Functional Dependency (FD)

2. Inference Problem

3. Closure Algorithm

Functional Dependency

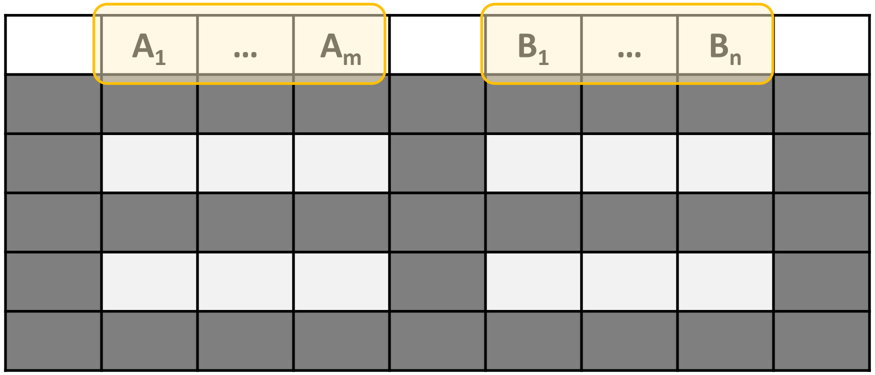
Def: Let A,B be *sets* of attributes
We write A → B or say A *functionally determines* B if, for any tuples t_1 and t_2:

$$t_1[A] = t_2[A] \text{ implies } t_1[B] = t_2[B]$$

and we call A → B a functional dependency

A->B means that
“whenever two tuples agree on A then they agree on B.”

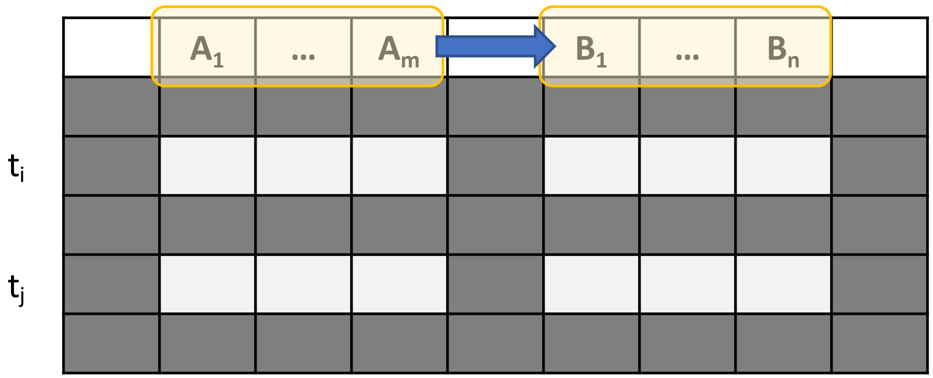
A Picture Of FDs



Defn (again):

Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

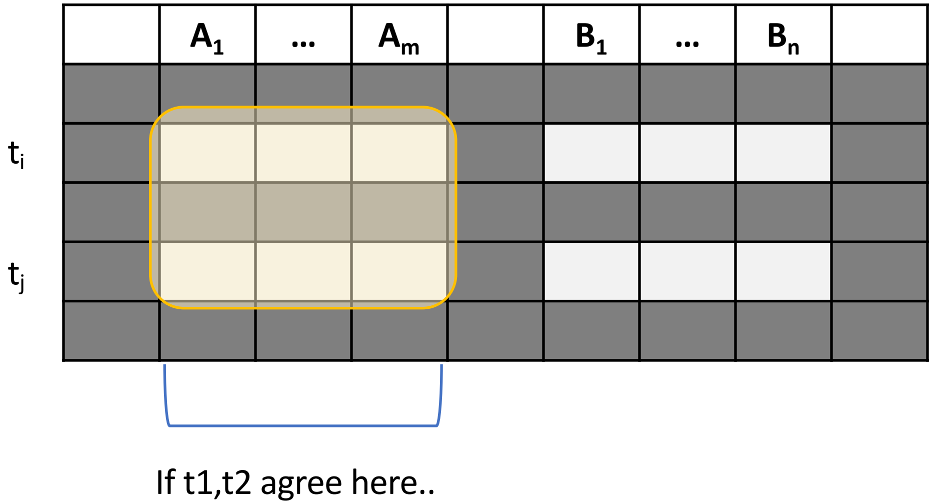
A Picture Of FDs



Defn (again):
Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

The *functional dependency* A→ B on R holds if for ***any*** t_i,t_j in R:

A Picture Of FDs

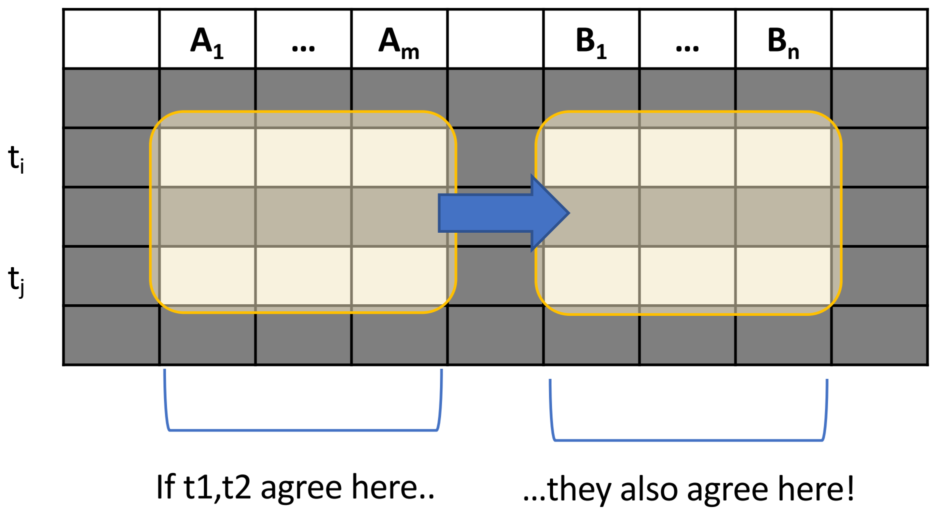


<u>Defn (again):</u>
Given attribute sets $A=\{A_1,\ldots,A_m\}$ and $B = \{B_1,\ldots B_n\}$ in R,

The *functional dependency* $A \rightarrow B$ on R holds if for ***any*** t_i,t_j in R:

$t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

A Picture Of FDs



Defn (again):

Given attribute sets **A={A_1,...,A_m}** and **B = {B_1,...B_n}** in **R**,

The *functional dependency* A→ B on **R** holds if for ***any*** t_i, t_j in R:

if $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND ... AND $t_i[A_m] = t_j[A_m]$

then $t_i[B_1] = t_j[B_1]$ AND $t_i[B_2]=t_j[B_2]$ AND ... AND $t_i[B_n] = t_j[B_n]$

Example

An FD holds, or does not hold on a table:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

√ Position → Phone

x Phone → Position

√ Phone, Name → Position

Exercise - 1

An FD <u>holds</u>, or <u>does not hold</u> on a table:

Name	Category	Color	Department	Price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	49
Gizmo	Stationary	Green	Office-supply	59

√ 1. Name → Color

√ 2. Category → Department

√ 3. Color, Category → Color

Exercise - 2

A	B	C	D	E
1	2	4	3	6
3	2	5	1	8
1	4	4	5	7
1	2	4	3	6
3	2	5	1	8

Find at least *three* FDs which **do not hold** on this table:

{ } → { }
{ } → { }
{ } → { }

Outline

1. Functional Dependency (FD)

2. **Inference Problem**

3. Closure Algorithm

An Interesting Observation

Provided FDs:

1. Name → Color
2. Category → Department
3. Color, Category → Price

Does it always hold? **Name, Category → Price**

If we find out from application domain that a relation satisfies some FDs, it doesn't mean that we found all the FDs that it satisfies! There could be more FDs implied by the ones we have

Inference Problem

Whether or not a set of FDs imply another FD?

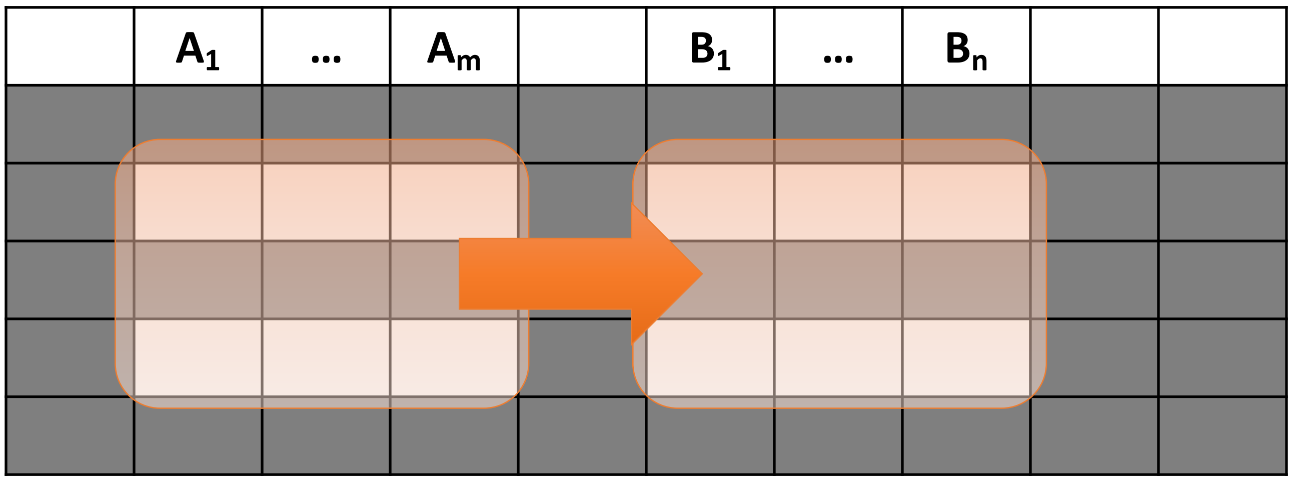
This is called **Inference problem**

Answer: Three simple rules called
Armstrong's Rules.

1. Split/Combine,
2. Reduction, and
3. Transitivity

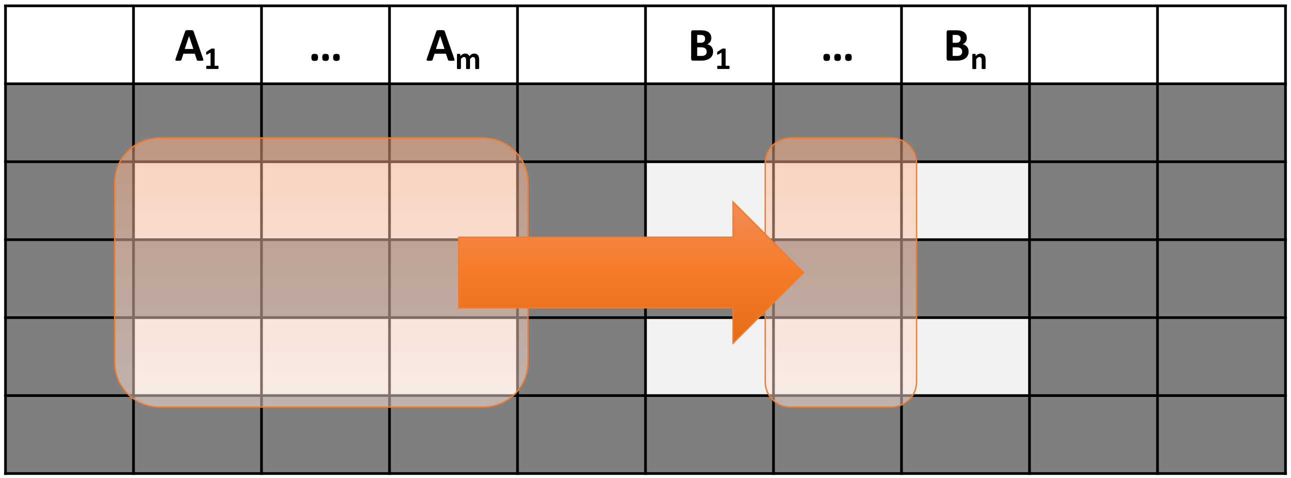
William Ward Armstrong is a Canadian mathematician and computer scientist. He earned his Ph.D. from the University of British Columbia in 1966 and is most known as the originator Armstrong's axioms of dependency in a Relational database.[1]

1. Split/Combine



$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$

1. Split/Combine

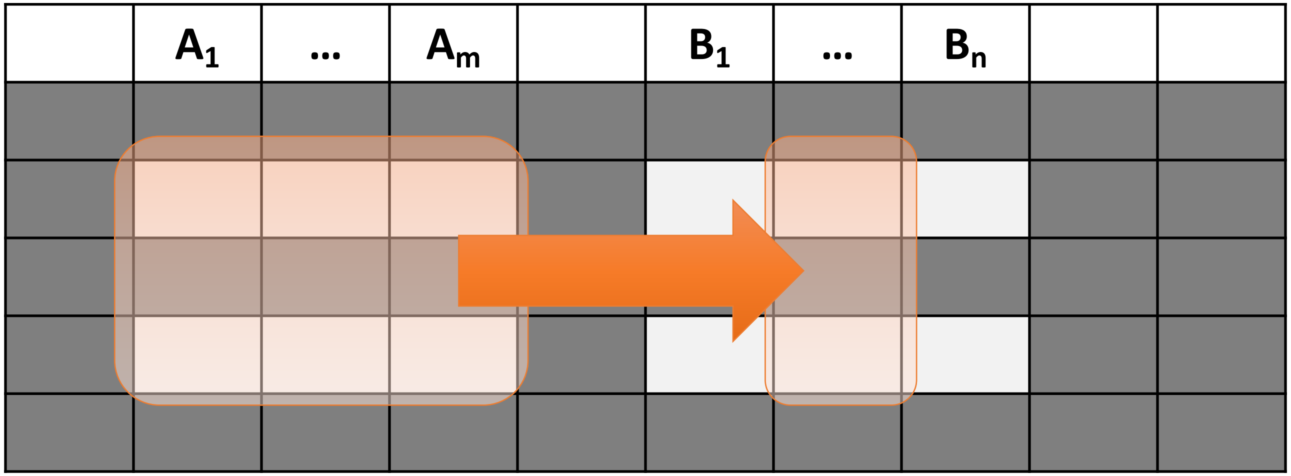


$$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$$

… is equivalent to the following *n* FDs…

$$A_1,\ldots,A_m \rightarrow B_i \text{ for } i=1,\ldots,n$$

1. Split/Combine

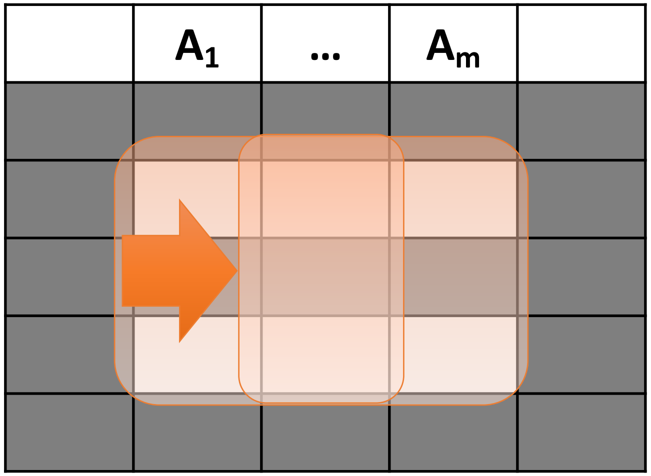


And vice-versa, $A_1,\ldots,A_m \rightarrow B_i$ for $i=1,\ldots,n$

… is equivalent to …

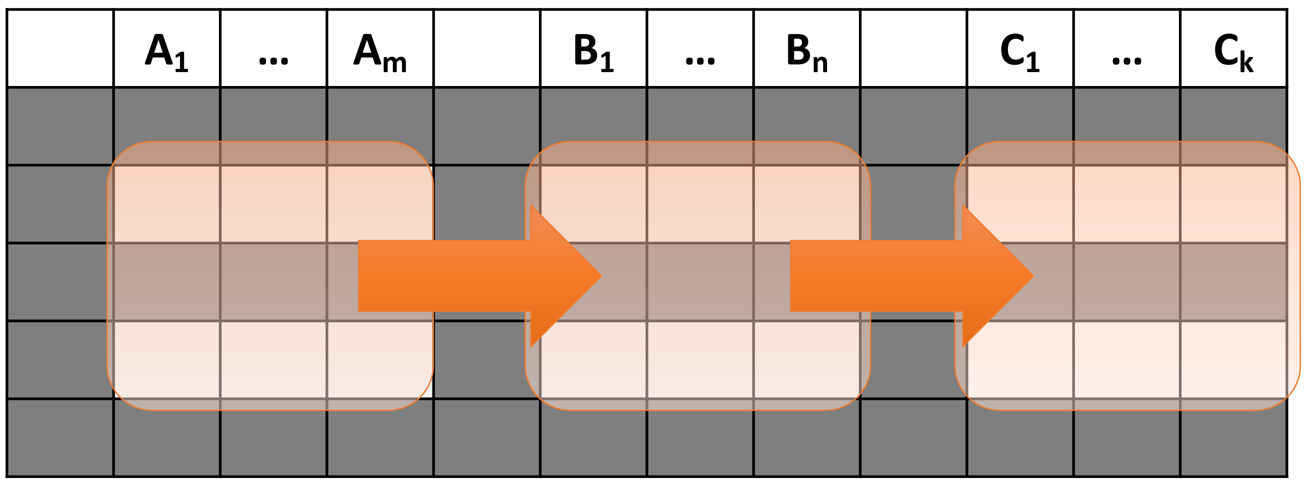
$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$

2. Reduction/Trivial



$A_1,\ldots,A_m \rightarrow A_j$ for any $j=1,\ldots,m$

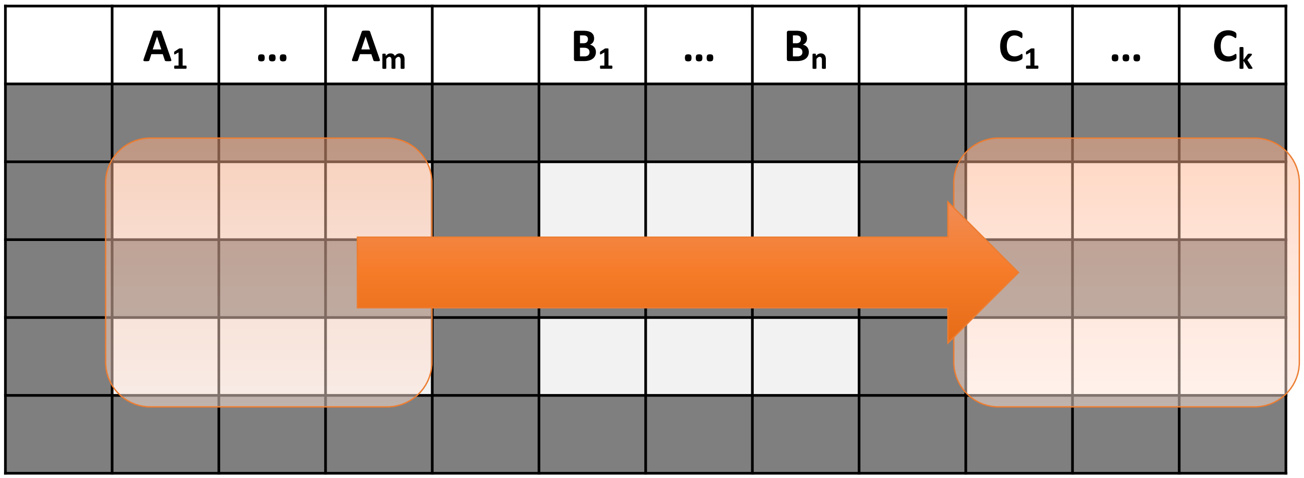
3. Transitive



$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$ and

$B_1,\ldots,B_n \rightarrow C_1,\ldots,C_k$

3. Transitive



$A_1, \ldots, A_m \rightarrow B_1,\ldots,B_n$ and
$B_1,\ldots,B_n \rightarrow C_1,\ldots,C_k$

implies

$A_1,\ldots,A_m \rightarrow C_1,\ldots,C_k$

Inferred FDs

Example:

Inferred FDs:

Inferred FD	Rule used
4. Name, Category → Name	?
5. Name, Category → Color	?
6. Name, Category → Category	?
7. Name, Category → Color, Category	?
8. Name, Category → Price	?

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Which FDs hold?

Inferred FDs

Example:

Inferred FDs:

Inferred FD	Rule used
4. Name, Category → Name	Trivial
5. Name, Category → Color	Transitive (4 -> 1)
6. Name, Category → Category	Trivial
7. Name, Category → Color, Category	Split/combine (5 + 6)
8. Name, Category → Price	Transitive (7 -> 3)

Provided FDs:

1. {Name} → {Color}
2. {Category} → {Dept.}
3. {Color, Category} → {Price}

Can we find an algorithmic way to do this?

Outline

1. Functional Dependency (FD)

2. Inference Problem

3. **Closure Algorithm**

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$ and a set of FDs **F**:

Then the closure, $\{A_1, \ldots, A_n\}^+$ is the set of attributes B s.t. $\{A_1, \ldots, A_n\} \rightarrow B$

Example: F =

```
name → color
category → department
color, category → price
```

Closures:

```
{name}+ = {name, color}
{name, category}+ = {name, category, color, dept, price}
{color}+ = {color}
```

Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$ and set of FDs F.

Repeat until X doesn't change; **do**:

if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F

and $\{B_1, \ldots, B_n\} \subseteq X$

then add C to X.

Return X as X^+

Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

```
{name, category}⁺ =
{name, category}
```

F =

```
name → color

category → dept

color, category → price
```

Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

`{name, category}`$^+$ =
`{name, category}`

`{name, category}`$^+$ =
`{name, category, color}`

F =

```
name → color

category → dept

color, category → price
```

Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
and $\{B_1, \ldots, B_n\} \subseteq X$:
then add C to X.
Return X as X^+

```
{name, category}+ =
{name, category}
```

```
{name, category}+ =
{name, category, color}
```

```
{name, category}+ =
{name, category, color, dept}
```

F =

```
name → color
category → dept
color, category → price
```

Closure Algorithm

Start with $X = \{A_1, \ldots, A_n\}$, FDs F.
Repeat until X doesn't change; **do**:
  if $\{B_1, \ldots, B_n\} \rightarrow C$ is in F
    and $\{B_1, \ldots, B_n\} \subseteq X$:
      then add C to X.
Return X as X^+

F =

```
name → color

category → dept

color, category → price
```

```
{name, category}+ =
{name, category}
```

```
{name, category}+ =
{name, category, color}
```

```
{name, category}+ =
{name, category, color, dept}
```

```
{name, category}+ =
{name, category, color, dept,
price}
```

Exercise - 3

```
R(A,B,C,D,E,F)
```

```
A,B → C
A,D → E
  B → D
A,F → B
```

Compute $\{A,B\}^+ = \{A, B,$ $\}$

Compute $\{A, F\}^+ = \{A, F,$ $\}$

Exercise - 3

R(A,B,C,D,E,F)

A,B → C
A,D → E
B → D
A,F → B

Compute $\{A,B\}^+ = \{A, B, C, D \qquad \}$

Compute $\{A, F\}^+ = \{A, F, B \qquad \}$

Exercise - 3

```
R(A,B,C,D,E,F)
```

```
A,B → C
A,D → E
  B → D
A,F → B
```

Compute $\{A,B\}^+ = \{A, B, C, D, E\}$

Compute $\{A, F\}^+ = \{A, B, C, D, E, F\}$

Exercise - 4

- Find all FD's implied by

```
A,B → C
A,D → B
B   → D
```

Requirements

1. **Non-trivial** FD (i.e., no need to return A, B → A)
2. The right-hand side contains **a single** attribute (i.e., no need to return A, B → C, D)

Exercise - 4

Given F =

$$A,B \rightarrow C$$
$$A,D \rightarrow B$$
$$B \rightarrow D$$

Step 1: Compute X^+, for every set of attributes X:

$\{A\}^+ = ?$
$\{B\}^+ = ?$
$\{C\}^+ = ?$
$\{D\}^+ = ?$
$\{A,B\}^+ = ?$
$\{A,C\}^+ = ?$
$\{A,D\}^+ = ?$
$\{B,C\}^+ = ?$
$\{B,D\}^+ = ?$
$\{C,D\}^+ = ?$
$\{A,B,C\}^+ = ?$
$\{A,B,D\}^+ = ?$
$\{A,C,D\}^+ = ?$
$\{B,C,D\}^+ = ?$
$\{A,B,C,D\}^+ = ?$

Exercise - 4

Given F =

$$A,B \rightarrow C$$
$$A,D \rightarrow B$$
$$B \rightarrow D$$

Step 1: Compute X^+, for every set of attributes X:

$\{A\}^+ = \{A\}$
$\{B\}^+ = \{B,D\}$
$\{C\}^+ = \{C\}$
$\{D\}^+ = \{D\}$
$\{A,B\}^+ = \{A,B,C,D\}$
$\{A,C\}^+ = \{A,C\}$
$\{A,D\}^+ = \{A,B,C,D\}$
$\{B,C\}^+ = \{B,C,D\}$
$\{B,D\}^+ = \{B,D\}$
$\{C,D\}^+ = \{C,D\}$
$\{A,B,C\}^+ = \{A,B,C,D\}$
$\{A,B,D\}^+ = \{A,B,C,D\}$
$\{A,C,D\}^+ = \{A,B,C,D\}$
$\{B,C,D\}^+ = \{B,C,D\}$
$\{A,B,C,D\}^+ = \{A,B,C,D\}$

Exercise - 4

Given F =

```
A,B → C
A,D → B
B   → D
```

Step 2: Enumerate all FDs X → Y, s.t. Y ⊆ X^+ and X ∩ Y = ∅:

```
{A}+ = {A}
{B}+ = {B,D}
{C}+ = {C}
{D}+ = {D}
{A,B}+ = {A,B,C,D}
{A,C}+ = {A,C}
{A,D}+ = {A,B,C,D}
{B,C}+ = {B,C,D}
{B,D}+ = {B,D}
{C,D}+ = {C,D}
{A,B,C}+ = {A,B,C,D}
{A,B,D}+ = {A,B,C,D}
{A,C,D}+ = {A,B,C,D}
{B,C,D}+ = {B,C,D}
{A,B,C,D}+ = {A,B,C,D}
```

```
B → D
A,B → C
A,B → D
A,D → B
A,D → C
B,C → D
A,B,C → D
A,B,D → C
A,C,D → B
```

Review

1. Functional Dependency (FD)
   - What is an FD?

2. Inference Problem
   - Whether or not a set of FDs imply another FD?

3. Closure
   - How to compute the closure of attributes?

High-level Idea

Student	Course	Room
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

Student	Course
Mike	354
Mary	354
Sam	354
..	..

Course	Room
354	AQ3149
454	T9204

Two Steps

1. Search for “bad” FDs in the table
2. *Keep decomposing the table into sub-tables* until no more bad FDs

Like a debugging process ☺

Outline

- “Good” vs. “Bad” FDs
- Boyce-Codd Normal Form
- Decompositions

"Good" vs. "Bad" FDs

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID → Name, Phone, Position

Good FD since EmpID can determine everything

EmpID is a Key

Position → Phone

Bad FD since Phone cannot determine everything

Exercise - 1

Student	Course	Room
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

Student, Course → Room   Good FD!

Course → Room   Bad FD!

What's wrong with "Bad" FDs

- If X →Y is a Bad FD, then X functionally determines *some* of the attributes; therefore, those other attributes can be duplicated
  - Recall: this means there is redundancy
  - And redundancy like this can lead to data anomalies!

Student	**Course**	**Room**
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

Outline

- “Good” vs. “Bad” FDs
- Boyce-Codd Normal Form
- Decompositions

Boyce-Codd Normal Form (BCNF)

- Main idea is that we define "good" and "bad" FDs as follows:

  - X → A is a "*good FD" if X is a key*
    - In other words, if A is the set of all attributes

  - X → A is a *"bad FD"* otherwise

- We will try to eliminate the "bad" FDs!

Boyce-Codd Normal Form (BCNF)

A relation R is **<u>in BCNF</u>** if:
there are no "bad" FDs

A relation R is <u>in BCNF</u> if:

if $\{A_1, \ldots, A_n\} \rightarrow B$ is a *non-trivial* FD in R

then $\{A_1, \ldots, A_n\}$ **is a key** for R

Equivalently: $\forall$ sets of attributes X, either ($X^+ = X$) or (X^+ = all attributes)

Example

Is this table in BCNF?

Name	SIN	PhoneNumber	City
Fred	123-45-6789	604-555-1234	Vancouver
Fred	123-45-6789	604-555-6543	Vancouver
Joe	987-65-4321	908-555-2121	Burnaby
Joe	987-65-4321	908-555-1234	Burnaby

`{SIN} → {Name,City}`

This FD is *bad* because it is <u>not</u> a key

⟹ <u>Not</u> in BCNF

What is the key?
{SIN, PhoneNumber}

Example

Name	SIN	City
Fred	123-45-6789	Vancouver
Joe	987-65-4321	Burnaby

SIN	PhoneNumber
123-45-6789	604-555-1234
123-45-6789	604-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

`{SIN} → {Name,City}`

This FD is now *good* because it is the key

Now in BCNF!

BCNF Decomposition Algorithm

BCNFDecomp(R):

BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X* $\rightarrow$ *Y*

X is not a key, i.e.,
$X^+ \neq$ [all attributes]

BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X → Y*

  <u>if</u> (not found) <u>then</u> Return R

If no “bad” FDs found, in BCNF!

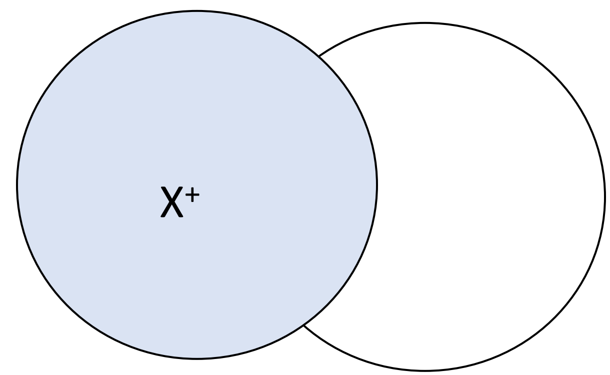
BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X → Y*

  if (not found) then **Return** R

  Split R into X^+ and X+[rest attributes]

One table is X^+



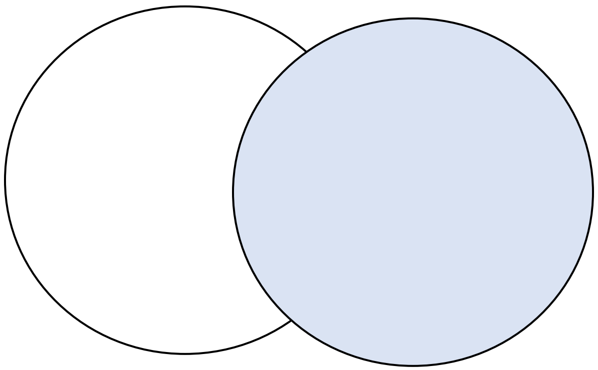
BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X → Y*

  <u>if</u> (not found) <u>then</u> **Return** R

  Split R into X^+ and **X+[rest attributes]**

The other table is
$X + (R - X^+)$



BCNF Decomposition Algorithm

BCNFDecomp(R):
  Find *a non-trivial bad FD: X → Y*

  if (not found) **then** **Return** R

  Split R into X^+ and X+[rest attributes]

  Return BCNFDecomp(R_1), BCNFDecomp(R_2)

Proceed recursively until no more “bad” FDs!

BCNF Decomposition Algorithm

BCNFDecomp(R):
Find *a non-trivial bad FD: X → Y*

<u>if</u> (not found) **<u>then</u> Return** R

Split R into X^+ and X+[rest attributes]

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

Only look at the FD in the given set

Need to imply all FDs for R_1 and R_2

Example

Student	Course	Room
Mike	354	AQ3149
Mary	354	AQ3149
Sam	354	AQ3149
..	..	..

Course → Room

Student	Course
Mike	354
Mary	354
Sam	354
..	..

Course	Room
354	AQ3149
454	T9204

Exercise - 2

BCNFDecomp(R):

Find *a non-trivial bad FD: X → Y*

<u>if</u> (not found) **<u>then</u> Return** R

Split R into X^+ and X+[rest attributes]

Return BCNFDecomp(R_1), BCNFDecomp(R_2)

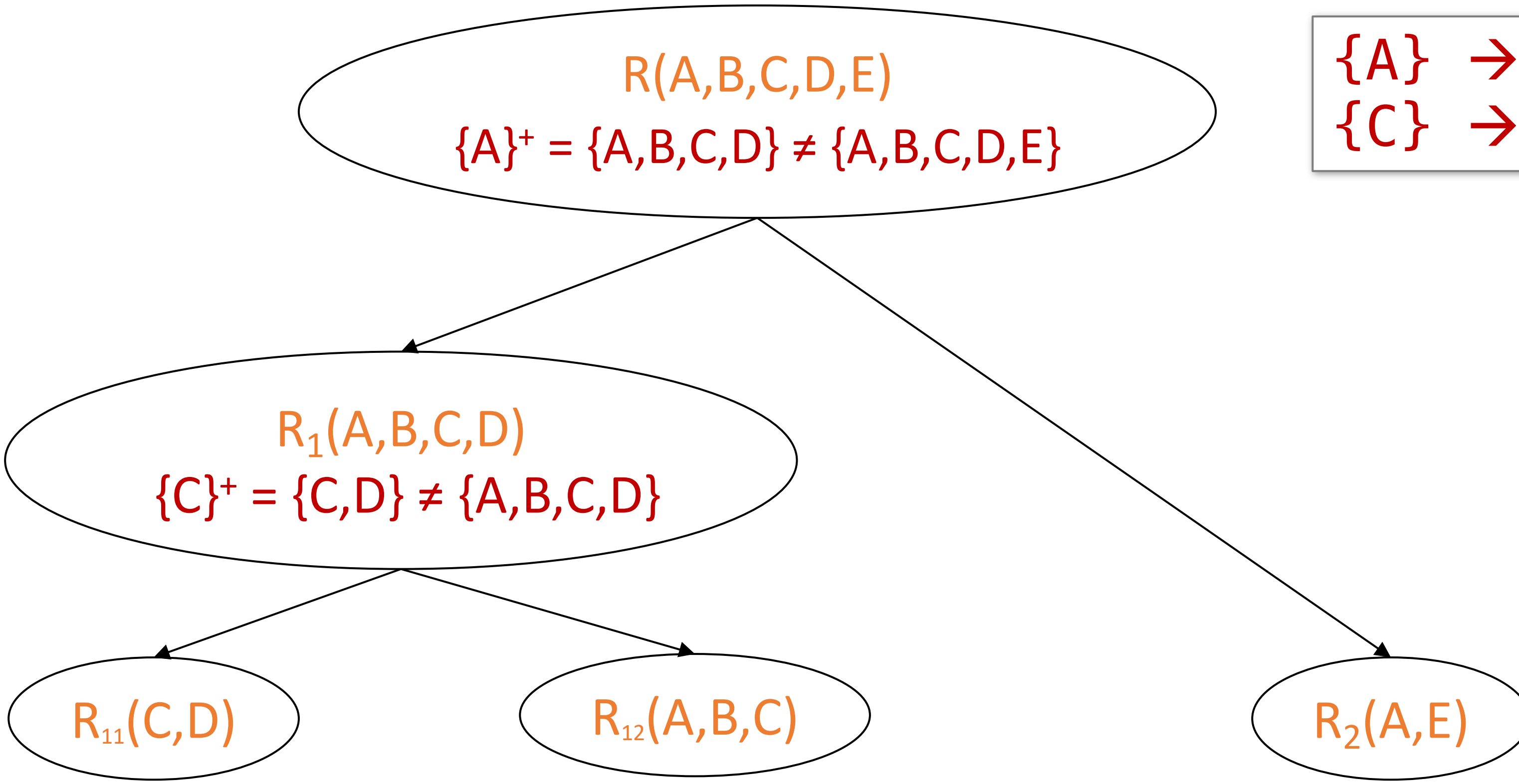
R(A,B,C,D,E)

{A} → {B,C}
{C} → {D}

Exercise - 2

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

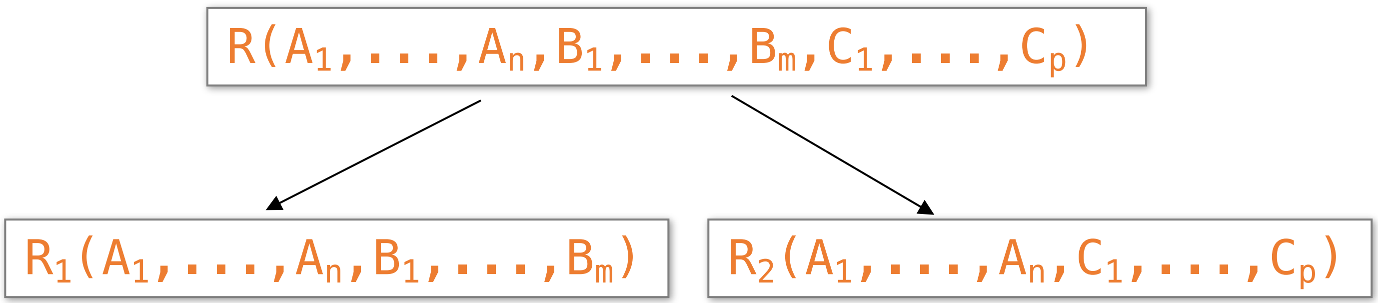
{A} → {B,C}
{C} → {D}



Outline

- “Good” vs. “Bad” FDs
- Boyce-Codd Normal Form
- Decompositions

Decompositions in General



R_1 = the *projection* of R on A_1, ..., A_n, B_1, ..., B_m

R_2 = the *projection* of R on A_1, ..., A_n, C_1, ..., C_p

Lossless Decompositions

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Price
Gizmo	19.99
OneClick	24.99
~~Gizmo~~	~~19.99~~

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

It is a <u>Lossless decomposition</u>

Lossless Decompositions

$R(A_1,\dots,A_n,B_1,\dots,B_m,C_1,\dots,C_p)$

$R_1(A_1,\dots,A_n,B_1,\dots,B_m)$

$R_2(A_1,\dots,A_n,C_1,\dots,C_p)$

A decomposition R to (R1, R2) is <u>lossless</u> if R = R1 Join R2

Lossy Decomposition

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

However sometimes it isn't

What's wrong here?

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossless Decompositions

$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

$R_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$

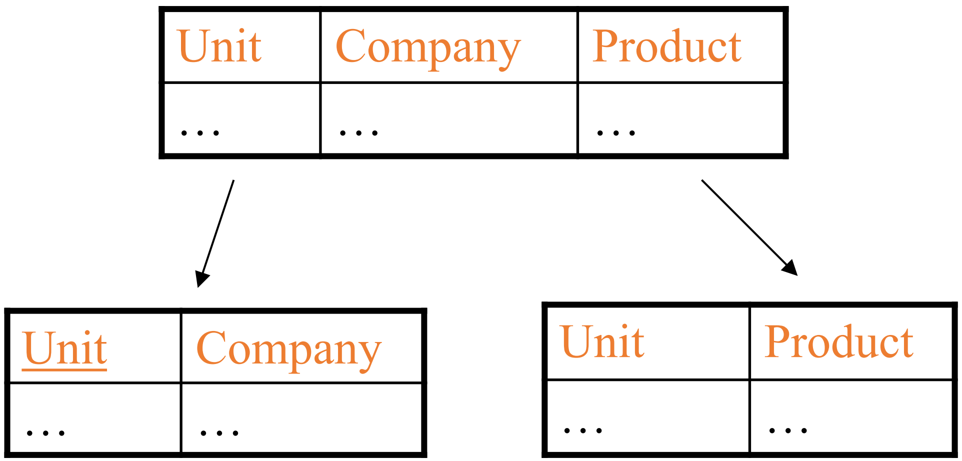
$R_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

If $\{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\}$
Then the decomposition is lossless

Note: don't need
$\{A_1, \ldots, A_n\} \rightarrow \{C_1, \ldots, C_p\}$

BCNF decomposition is always lossless.

A Problem with BCNF



{Unit} → {Company}
{Company,Product} → {Unit}

We do a BCNF decomposition on a “bad” FD:
$\{Unit\}^+ = \{Unit, Company\}$

{Unit} → {Company}

We lose the FD {Company,Product} → {Unit}!!

The Problem

- We started with a table R and FDs F
- We decomposed R into BCNF tables R_1, R_2, …
  with their own FDs F_1, F_2, …
- We insert some tuples into each of the relations—which satisfy their local FDs but when reconstruct it violates some FD **across** tables!

<u>Practical Problem</u>: To enforce FD, must reconstruct R—*on each insert!*

Trade-offs

- **Different Normal Forms**

Prevent Decomposition Problems

VS

Remove Redundancy

BCNF still most common- with additional steps to keep track of lost FDs…

Summary

- “Good” vs. “Bad” FDs
- Boyce-Codd Normal Form
- Decompositions

Acknowledge

- Some lecture slides were copied from or inspired by the following course materials
  - “W4111: Introduction to databases” by Eugene Wu at Columbia University
  - “CSE344: Introduction to Data Management” by Dan Suciu at University of Washington
  - “CMPT354: Database System I” by John Edgar at Simon Fraser University
  - “CS186: Introduction to Database Systems” by Joe Hellerstein at UC Berkeley
  - “CS145: Introduction to Databases” by Peter Bailis at Stanford
  - “CS 348: Introduction to Database Management” by Grant Weddell at University of Waterloo