

CSC 261/461 – Database Systems

Lecture 11

Spring 2018

Announcement

- Read the textbook!
 - Chapter 8:
 - Will cover later; But self-study the chapter
 - Section 8.1 – 8.5
 - Chapter 14:
 - Section 14.1 – 14.5
 - Chapter 15:
 - Section 15.1 – 15.4

Agenda

1. Finding functional dependencies
2. Closures, superkeys & keys

FINDING FUNCTIONAL DEPENDENCIES

What you will learn about in this section

1. “Good” vs. “Bad” FDs: Intuition
2. Finding FDs
3. Closures

“Good” vs. “Bad” FDs

We can start to develop a notion of **good** vs. **bad** FDs:

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

Intuitively:

EmpID -> Name, Phone, Position is “good FD”

- ***Minimal redundancy, less possibility of anomalies***

“Good” vs. “Bad” FDs

We can start to develop a notion of **good** vs. **bad** FDs:

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

Intuitively:

EmpID → Name, Phone, Position is “good FD”

But Position → Phone is a “bad FD”

- **Redundancy!**
Possibility of data anomalies

“Good” vs. “Bad” FDs

| Student | Course | Room |
|---------|--------|------|
| Mary | CS145 | B01 |
| Joe | CS145 | B01 |
| Sam | CS145 | B01 |
| .. | .. | .. |

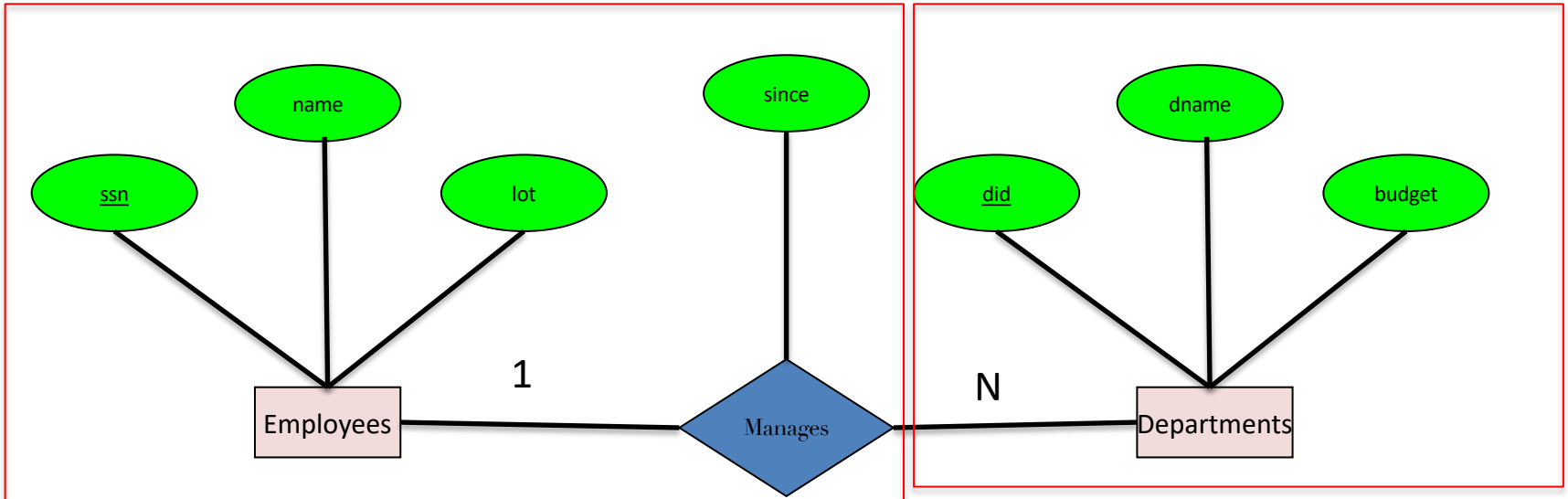
Returning to our original example... can you see how the “bad FD” {Course} -> {Room} could lead to an:

- Update Anomaly
- Insert Anomaly
- Delete Anomaly
- ...

Given a set of FDs (from user) our goal is to:

1. Find all FDs, and
2. Eliminate the “Bad Ones”.

Recall: What about this?



```
CREATE TABLE Employee_Mgr( ssn char(11),  
                             name varchar(30),  
                             lot integer,  
                             since date,  
                             did integer,
```

```
PRIMARY KEY (ssn, did),  
FOREIGN KEY (did) REFERENCES Departments (did)  
ON DELETE RESTRICT,  
)
```

Possible. But...

Our Next topic...
It's an example
of bad
Functional
Dependency

Why exactly is it bad?

- $ssn \rightarrow name, lot$
- $did \rightarrow ssn, since$
- Wow.. Now it looks like did is the primary key for Employee Relation !!!
- $did \rightarrow ssn, since, name, lot$

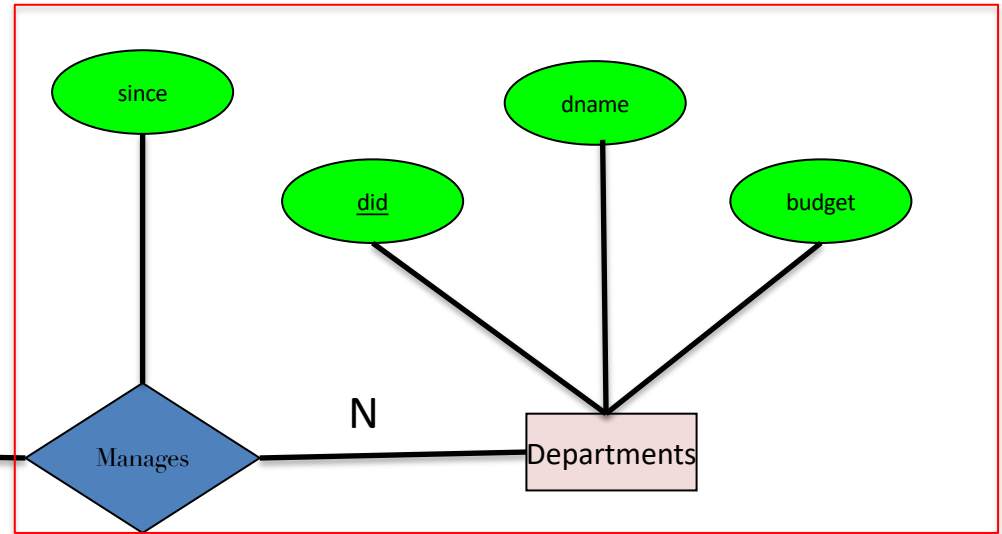
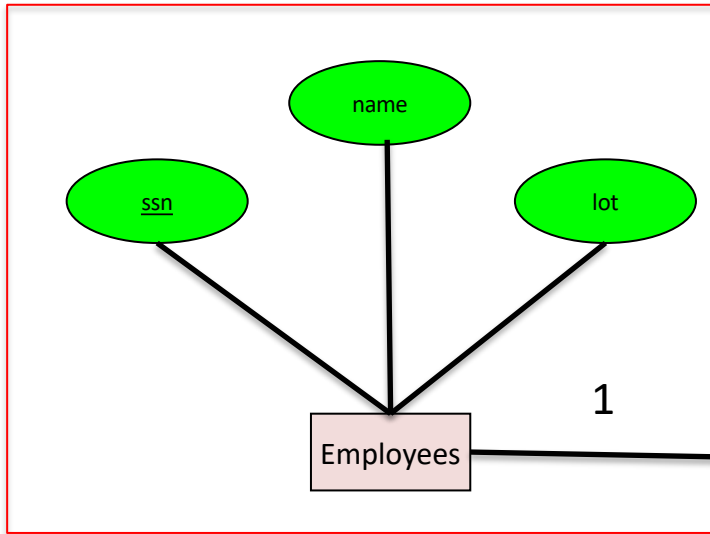
Transitive dependency.

$did \rightarrow ssn \rightarrow name$

We have to normalize this table!

Moreover, for making (ssn, did) as the primary key, we should have more departments than the number of employees (to avoid 'null' departments).
In real life, this will be almost never be the case.

Recall: Better way of doing it (Option 3)



```
CREATE TABLE Dept_Mgr (
```

```
did integer  
dname varchar(30),  
budget float(30),  
ssn char(11),  
since date,
```

```
PRIMARY KEY (did),
```

```
FOREIGN KEY (ssn) REFERENCES Employees ON DELETE SET NULL,
```

```
)
```

did → dname, budget, ssn, since
It's a good relation. No need to normalize it

FDs for Relational Schema Design

- High-level idea: **why do we care about FDs?**
 1. Start with some relational *schema*
 2. Find out its **functional dependencies (FDs)**
 3. Use these to **design a better schema**
 1. One which minimizes possibility of anomalies

Finding Functional Dependencies

- There can be a very **large number** of FDs...
 - How to find them all **efficiently**?
- We can't necessarily show that any FD will hold **on all instances...**
 - How to do this?

We will start with this problem:

Given a set of FDs, F , what other FDs ***must*** hold?

Finding Functional Dependencies

Equivalent to asking: Given a set of FDs, $F = \{f_1, \dots, f_n\}$, does an FD g hold?

Inference problem: How do we decide?

Finding Functional Dependencies

Example:

Products

| Name | Color | Category | Dep | Price |
|--------|-------|----------|--------|-------|
| Gizmo | Green | Gadget | Toys | 49 |
| Widget | Black | Gadget | Toys | 59 |
| Gizmo | Green | Whatsit | Garden | 99 |

Provided FDs:

1. {Name} \rightarrow {Color}
2. {Category} \rightarrow {Department}
3. {Color, Category} \rightarrow {Price}

Given the provided FDs, we can see that {Name, Category} \rightarrow {Price} must also hold on **any instance...**

Which / how many other FDs do?!?

Finding Functional Dependencies

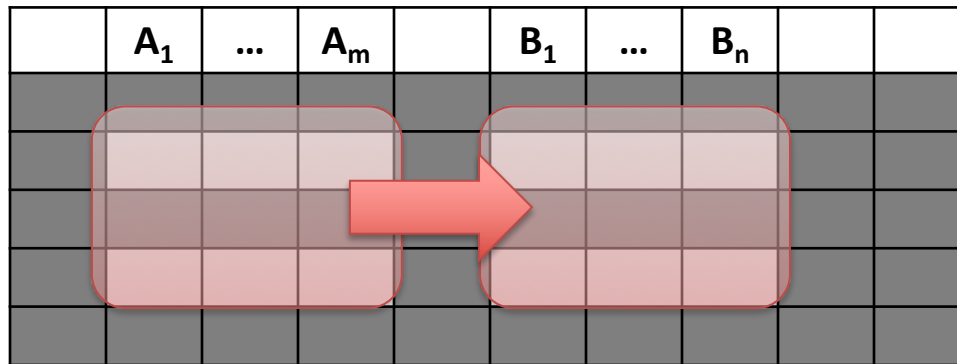
Equivalent to asking: Given a set of FDs, $F = \{f_1, \dots, f_n\}$, does an FD g hold?

Inference problem: How do we decide?

Answer: Three simple rules called
Armstrong's Rules.

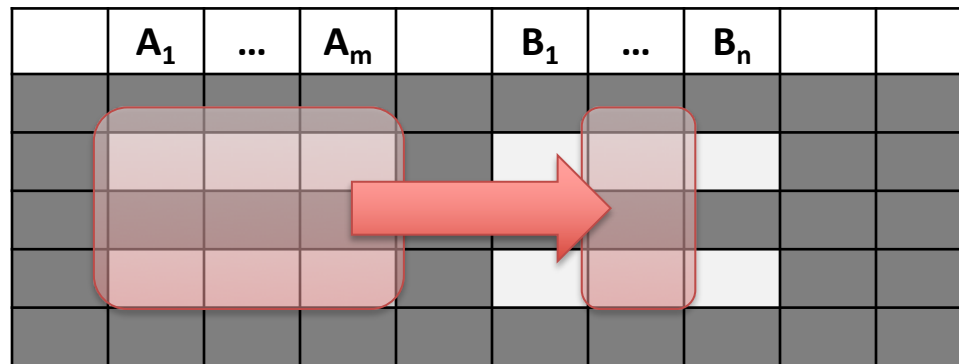
1. **Split/Combine,**
2. **Reduction, and**
3. **Transitivity... *ideas by picture***

1. Split/Combine (Decomposition & Union Rule)



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

1. Split/Combine (Decomposition & Union Rule)

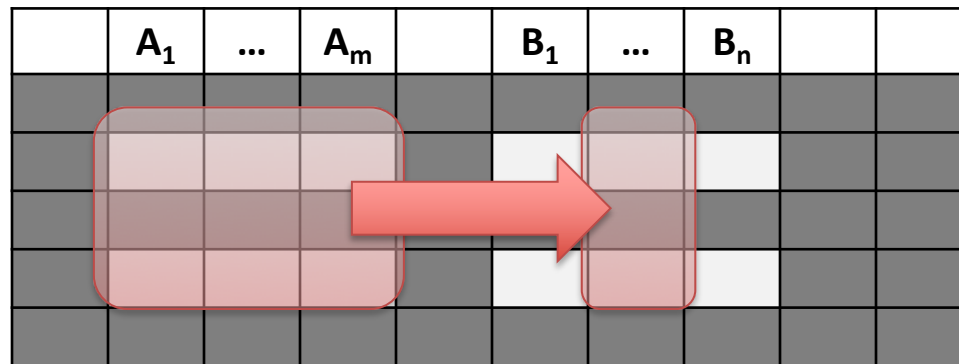


$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n$$

... is equivalent to the following n FDs...

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

1. Split/Combine (Decomposition & Union Rule)

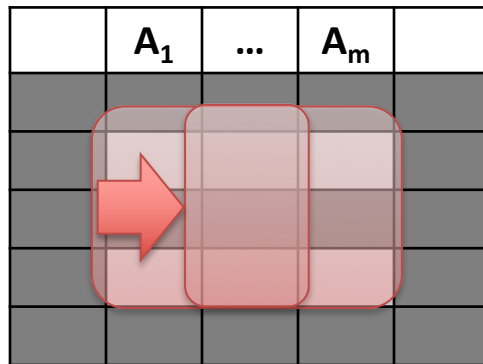


And vice-versa, $A_1, \dots, A_m \rightarrow B_i$ for $i=1, \dots, n$

... is equivalent to ...

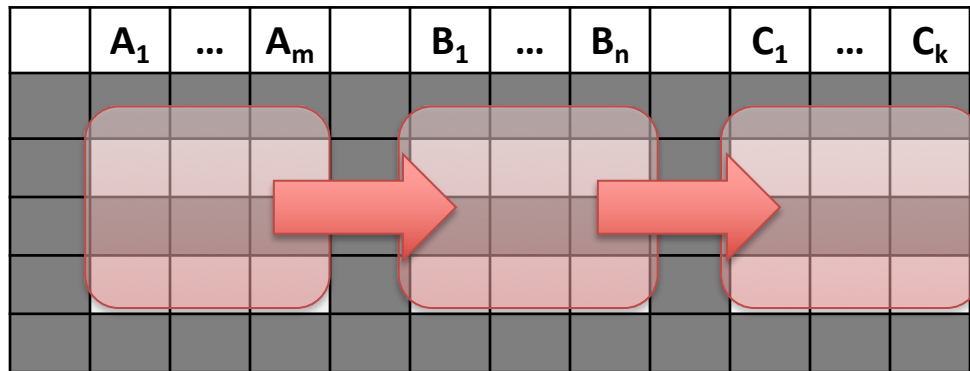
$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n$$

2. Reduction/Trivial (Reflexive Rule)



$$A_1, \dots, A_m \rightarrow A_j \text{ for any } j=1, \dots, m$$

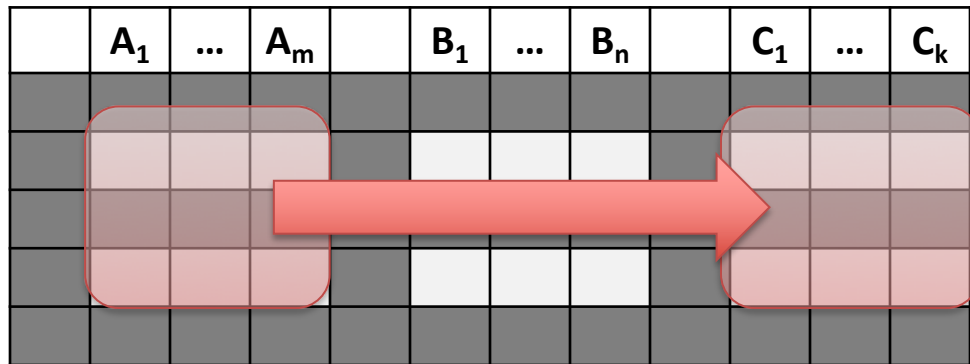
3. Transitive Rule



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

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

3. Transitive Rule



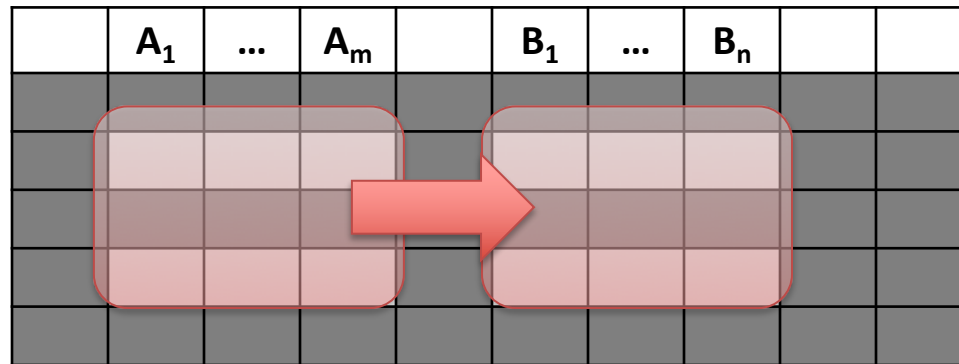
$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n \text{ and}$$

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

implies

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

Augmentation Rule



$A_1, \dots, A_m \rightarrow B_1, \dots, B_n$ implies

Augmentation Rule

| x_1 | A_1 | ... | A_m | | B_1 | ... | B_n | | |
|-------|-------|-----|-------|--|-------|-----|-------|--|--|
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

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

implies

$$x_1, A_1, \dots, A_m \rightarrow B_1, \dots, B_n$$

Finding Functional Dependencies

Example:

Products

| Name | Color | Category | Dep | Price |
|--------|-------|----------|--------|-------|
| Gizmo | Green | Gadget | Toys | 49 |
| Widget | Black | Gadget | Toys | 59 |
| Gizmo | Green | Whatsit | Garden | 99 |

Provided FDs:

1. {Name} \rightarrow {Color}
2. {Category} \rightarrow {Department}
3. {Color, Category} \rightarrow {Price}

Which / how many other FDs hold?

Finding Functional Dependencies

Example:

Provided FDs:

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

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} | ? |

Which / how many other FDs hold?

Finding Functional Dependencies

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?

Yes. But we need to learn about closures before that!

Closures

Closure of a set of Attributes

Given a set of attributes A_1, \dots, A_n and a set of FDs F :

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

Example: $F =$

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

**Example
Closures:**

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

Closure Algorithm

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

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

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

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

then add C to X .

Return X as X^+

Closure Algorithm

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

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

 $\{\text{category}\} \rightarrow \{\text{dept}\}$

 $\{\text{color, category}\} \rightarrow$
 $\{\text{price}\}$

Closure Algorithm

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

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color}\}$

Closure Algorithm

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

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept}\}$

Closure Algorithm

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

$F =$

$\{\text{name}\} \rightarrow \{\text{color}\}$

$\{\text{category}\} \rightarrow \{\text{dept}\}$

$\{\text{color, category}\} \rightarrow \{\text{price}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept}\}$

$\{\text{name, category}\}^+ =$
 $\{\text{name, category, color, dept, price}\}$

EXAMPLE

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

$\{A, B\} \rightarrow \{C\}$
 $\{A, D\} \rightarrow \{E\}$
 $\{B\} \rightarrow \{D\}$
 $\{A, F\} \rightarrow \{B\}$

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

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

EXAMPLE

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

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

$\{A, D\} \rightarrow \{E\}$

$\{B\} \rightarrow \{D\}$

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

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

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

EXAMPLE

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

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

$\{A, D\} \rightarrow \{E\}$

$\{B\} \rightarrow \{D\}$

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

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

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

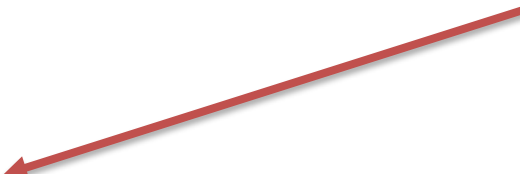
3. CLOSURES, SUPERKEYS & KEYS

What you will learn about in this section

1. Closures
2. Superkeys & Keys

Why Do We Need the Closure?

- With closure we can find all FD's easily
- To check if $X \rightarrow A$
 1. Compute X^+
 2. Check if $A \in X^+$



Note here that X is a *set* of attributes, but A is a *single* attribute. Why does considering FDs of this form suffice?

Recall the **Split/combine** rule:

$X \rightarrow A_1, \dots, X \rightarrow A_n$

implies

$X \rightarrow \{A_1, \dots, A_n\}$

Using Closure to Infer ALL FDs

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

Example:

Given $F =$

| | | |
|------------|---------------|-----|
| $\{A, B\}$ | \rightarrow | C |
| $\{A, D\}$ | \rightarrow | B |
| $\{B\}$ | \rightarrow | D |

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

No need to compute these- why?

We did not include $\{B, C\}$, $\{B, D\}$, $\{C, D\}$, $\{B, C, D\}$ to save some space.

Using Closure to Infer ALL FDs

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

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

Example:

Given $F =$

| | | |
|------------|---------------|-----|
| $\{A, B\}$ | \rightarrow | C |
| $\{A, D\}$ | \rightarrow | B |
| $\{B\}$ | \rightarrow | D |

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

Using Closure to Infer ALL FDs

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

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

Example:

Given $F =$

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

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

"Y is in the closure of X"

Using Closure to Infer ALL FDs

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

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

Example:

Given $F =$

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

Step 2: Enumerate all FDs $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

*The FD $X \rightarrow Y$
is non-trivial*

Superkeys and Keys

Keys and Superkeys

A **superkey** is a set of attributes A_1, \dots, A_n s.t. for *any other* attribute B in R , we have $\{A_1, \dots, A_n\} \rightarrow B$

I.e. all attributes are *functionally determined* by a superkey

A **key** is a *minimal* superkey

Meaning that no subset of a key is also a superkey

Finding Keys and Superkeys

- For each set of attributes X
 1. Compute X^+
 2. If $X^+ =$ set of all attributes then X is a **superkey**
 3. If X is minimal, then it is a **key**

Do we need to check all sets of attributes?

Example of Finding Keys

```
Product(name, price, category,  
color)
```

```
{name, category} → price  
{category} → color
```

What is a key?

Example of Keys

Product(name, price, category,
color)

{name, category} → price
{category} → color

$\{\text{name, category}\}^+ = \{\text{name, price, category, color}\}$

= the set of all attributes

⇒ this is a **superkey**

⇒ this is a **key**, since neither **name** nor **category** alone is a superkey

Acknowledgement

- Some of the slides in this presentation are taken from the slides provided by the authors.
- Many of these slides are taken from cs145 course offered by Stanford University.
- Thanks to YouTube, especially to [Dr. Daniel Soper](#) for his useful videos.