



DB Design

Prof. Hyuk-Yoon Kwon

<https://sites.google.com/view/seoultech-bigdata>

Most parts are based on slides used in Stanford (<http://web.stanford.edu/class/cs145>)

Contents

■ Summary of Previous Lecture

- Data manipulation
- View

■ Today's lecture

- DB Design

Today's Lecture

1. Normal forms & functional dependencies

- ACTIVITY: Finding FDs

2. Finding functional dependencies

3. Closures, superkeys & keys

- ACTIVITY: The key or a key?

1. Normal Forms & Functional Dependencies

What you will learn about in this section

- 1. Overview of design theory & normal forms**
- 2. Data anomalies & constraints**
- 3. Functional dependencies**
- 4. ACTIVITY: Finding FDs**

Design Theory

- Design theory is about how to represent your data to avoid *anomalies*.
- It is a mostly mechanical process
 - Tools can carry out routine portions

Normal Forms

- 1st Normal Form (1NF) = All tables are flat

- 2nd Normal Form = *disused*

- Boyce-Codd Normal Form (BCNF)

- 3rd Normal Form (3NF)

- *4th and 5th Normal Forms*

1st Normal Form (1NF)

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

Violates 1NF.

Student	Courses
Mary	CS145
Mary	CS229
Joe	CS145
Joe	CS106

In 1st NF

1NF Constraint: Types must be atomic!



Data Anomalies & Constraints

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes *anomalies*:

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

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

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes *anomalies*:

Student	Course	Room
Mary	CS145	B01
Joe	CS145	C12
Sam	CS145	B01
..

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

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes *anomalies*:

Student	Course	Room
..

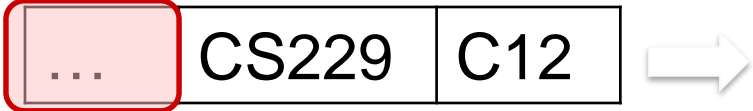
If everyone drops the class, we lose what room the class is in! = a **delete anomaly**

Constraints Prevent (some) Anomalies in the Data

A poorly designed database causes *anomalies*:

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

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



Constraints Prevent (some) Anomalies in the Data

Student	Course
Mary	CS145
Joe	CS145
Sam	CS145
..	..

Course	Room
CS145	B01
CS229	C12

Is this form better?

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

Today: develop theory to understand why this design may be better **and** how to find this *decomposition*...

Functional Dependencies

Functional Dependency

Def: Let A, B be sets of attributes

We write $A \rightarrow B$ or say A **functionally determines** B if, for any tuples t_1 and t_2 :

$t_1[A] = t_2[A]$ implies $t_1[B] = t_2[B]$

and we call $A \rightarrow B$ a **functional dependency**

$A \rightarrow B$ means that

“whenever two tuples agree on A then they agree on B .”

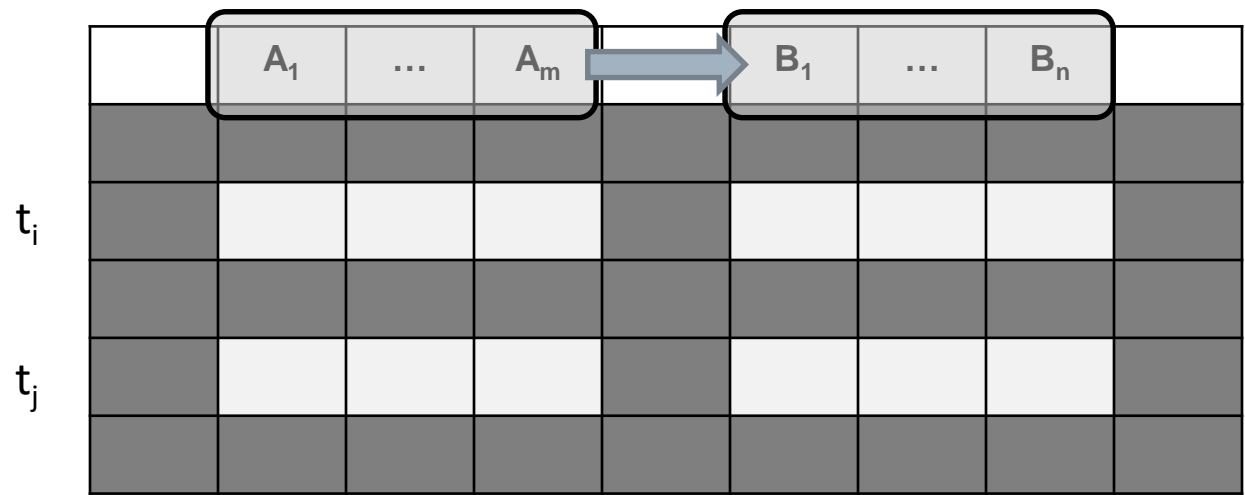
A Picture Of FDs

	A ₁	...	A _m		B ₁	...	B _n	

Defn (again):

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

A Picture Of FDs



Defn (again):

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

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

A Picture Of FDs

	A_1	...	A_m		B_1	...	B_n	
t_i								
t_j								

If t_1, t_2 agree here..

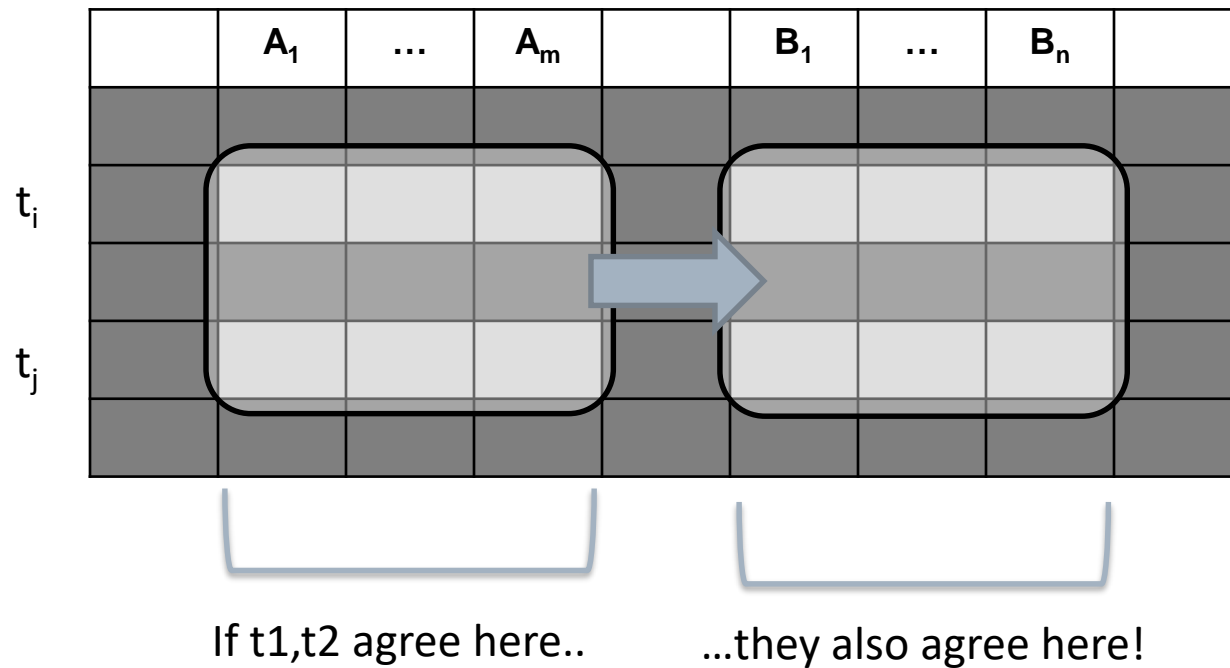
Defn (again):

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

The **functional dependency** $A \rightarrow B$ 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 $t_i[B_n] = t_j[B_n]$

Practice #1: Functional Dependency

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 are violated on this instance:

$\{ \} \rightarrow \{ \}$
 $\{ \} \rightarrow \{ \}$
 $\{ \} \rightarrow \{ \}$

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 the possibility of anomalies

Functional Dependencies as Constraints

A **functional dependency** is a form of **constraint**

- *Holds* on some instances (but not others) – can check whether there are violations
- Part of the schema, helps define a *valid* instance

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

Recall: an ***instance*** of a schema is a multiset of tuples conforming to that schema, ***i.e. a table***

Note: The FD {Course} \rightarrow {Room} ***holds on this instance***

Functional Dependencies as Constraints

Note that:

- You can check if an FD is **violated** by examining a single instance;
- However, you **cannot prove** that an FD is part of the schema by examining a single instance.
 - *This would require checking every valid instance*

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

However, cannot *prove* that the FD {Course} -> {Room} is ***part of the schema***

More Examples

An FD is a constraint which holds, or does not hold on an instance:

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

More Examples

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

{Position} → {Phone}

More Examples

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

but *not* {Phone} → {Position}

Practice #2: Functional Dependency with SQL

- Given the database schema $R(a, b, c)$, write an SQL query to test whether the functional dependency $b \rightarrow c$ holds on relation R .
 - You can check your SQL with ACDB databases mapping R into ACDB_CUSTOMERS, b into CUSTOMER_ID, and c into STATE

2. Finding Functional Dependencies

What you will learn about in this section

1. “Good” vs. “Bad” FDs: Intuition
2. Finding FDs
3. Closures
4. Practice: Compute the 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”
 $\{\text{Course}\} \rightarrow \{\text{Room}\}$ 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”.

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

This part can be tricky!

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

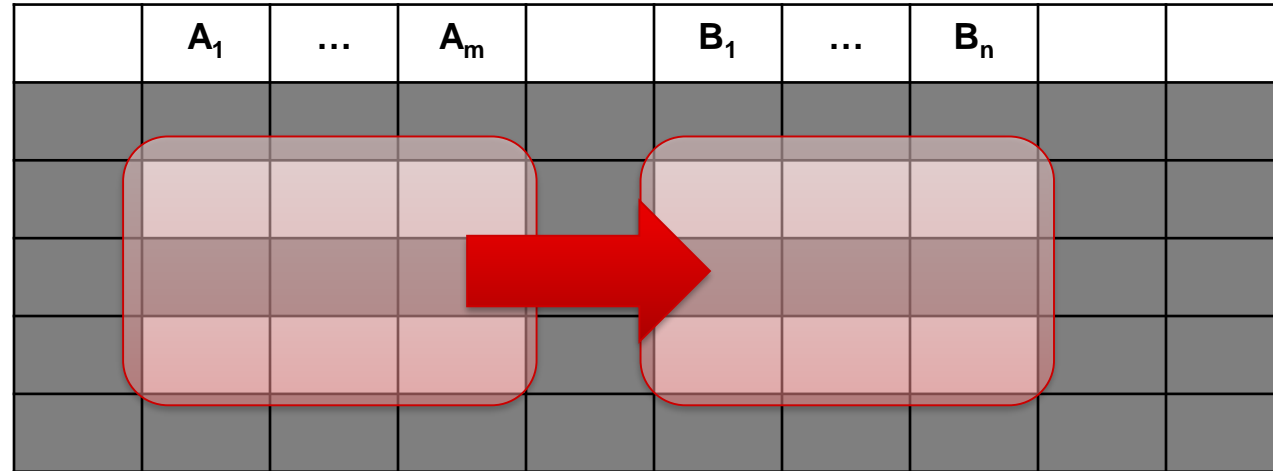
Equivalent to asking: Given a set of FDs, $F = \{f_1, \dots, f_n\}$, does an FD g hold?

Inference problem: How do we decide?

Answer: Three simple rules called
Armstrong's Rules.

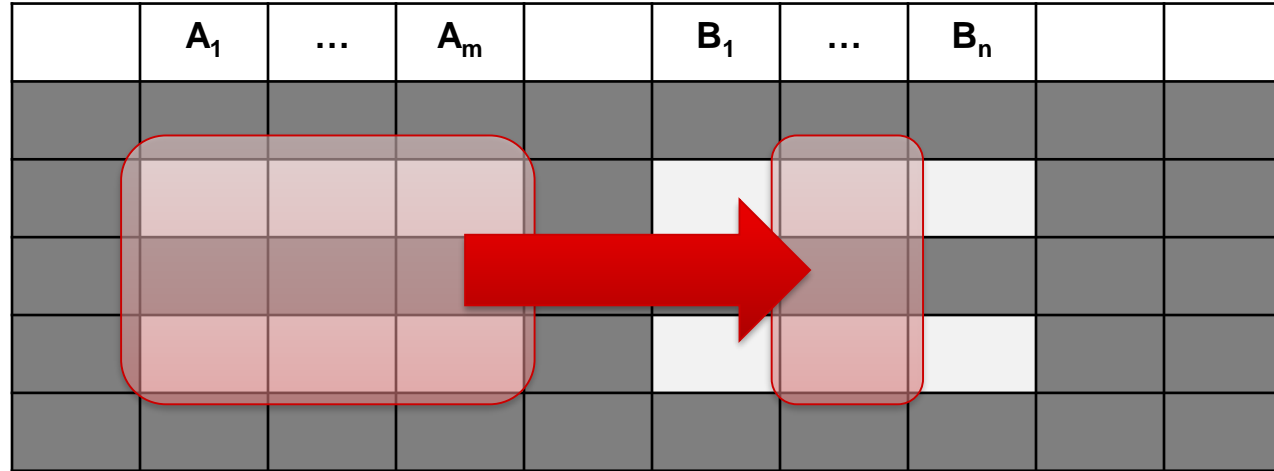
1. **Split/Combine,**
2. **Reduction, and**
3. **Transitivity...** *ideas by picture*

1. Split/Combine



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

1. Split/Combine

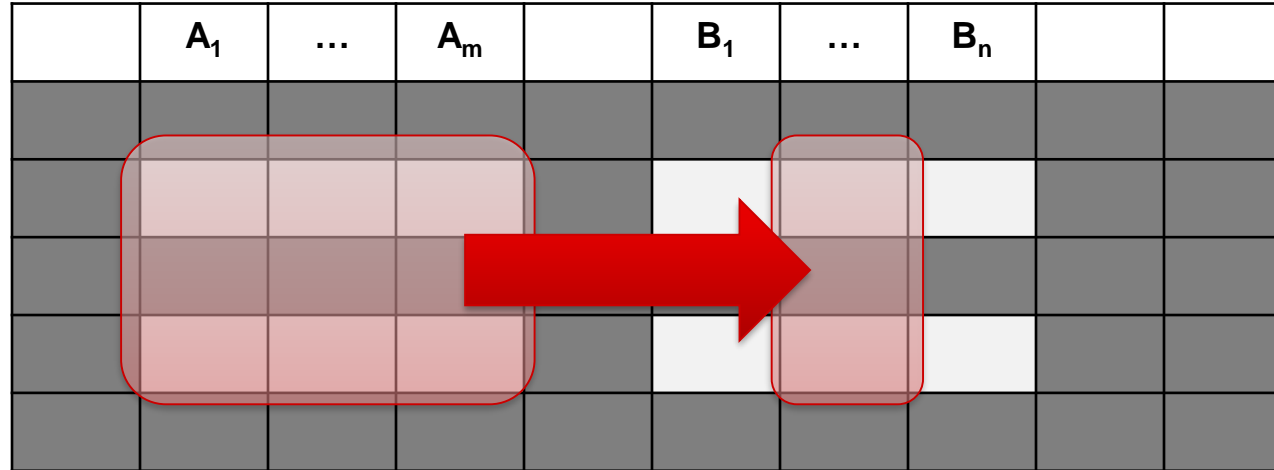


$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n$$

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

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

1. Split/Combine

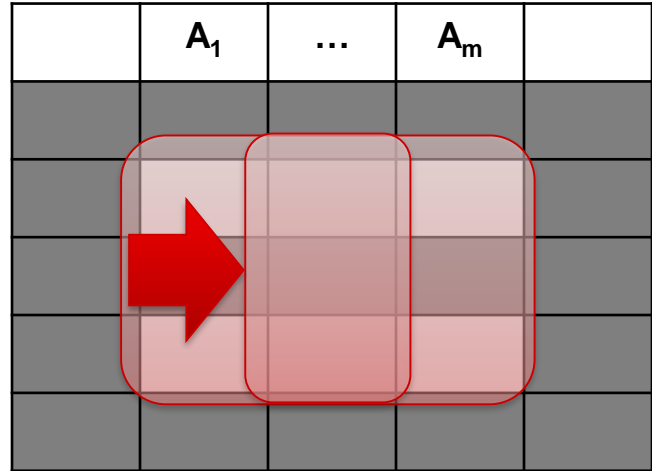


And vice-versa, $A_1, \dots, A_m \rightarrow B_i$ for $i=1, \dots, n$

... is equivalent to ...

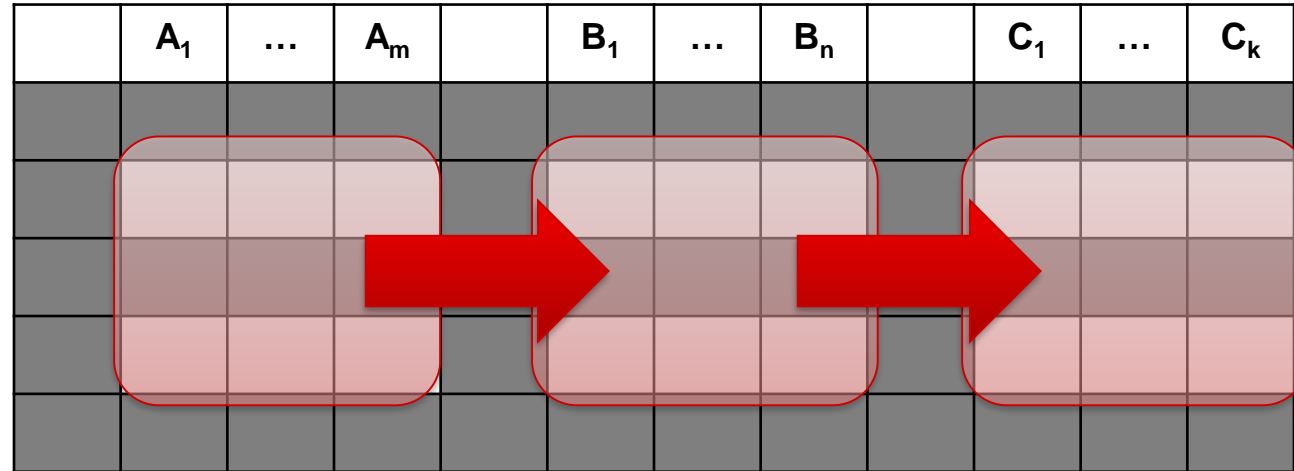
$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n$$

2. Reduction



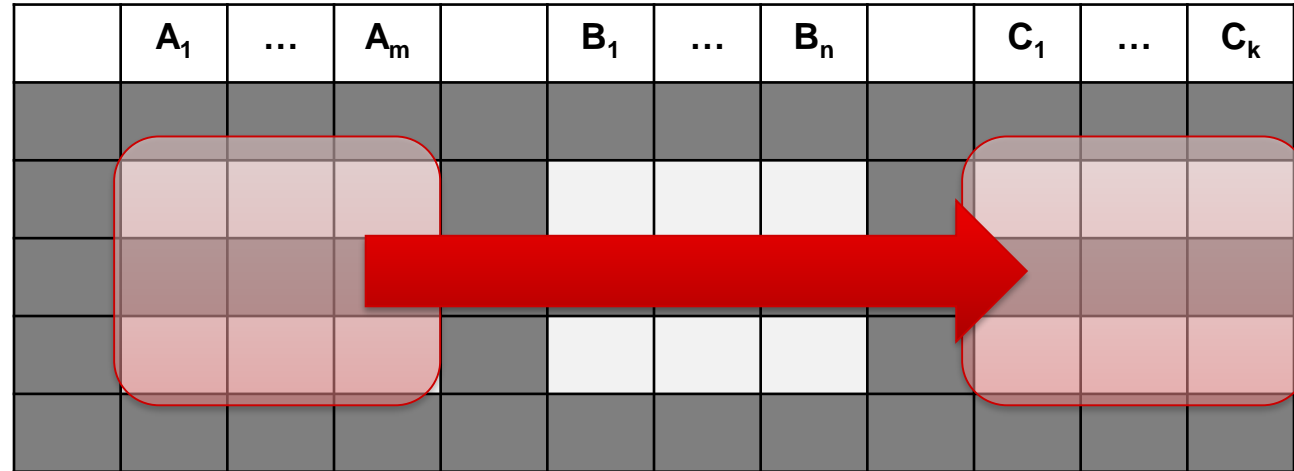
$$A_1, \dots, A_m \rightarrow A_j \text{ for any } j=1, \dots, m$$

3. Transitive Closure



$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n \text{ and} \\ B_1, \dots, B_n \rightarrow C_1, \dots, C_k$$

3. Transitive Closure



$$A_1, \dots, A_m \rightarrow B_1, \dots, B_n \text{ and} \\ B_1, \dots, B_n \rightarrow C_1, \dots, C_k$$

implies

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

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:

Inferred FDs:

Inferred FD	Rule used
4.	
5.	
6.	
7.	
8.	

Provided FDs:

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

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: $F =$

$$\begin{aligned} \{name\} &\rightarrow \{color\} \\ \{category\} &\rightarrow \{department\} \\ \{color, category\} &\rightarrow \{price\} \end{aligned}$$

***Example
Closures
:***

$$\begin{aligned} \{name\}^+ &= \{name, color\} \\ \{name, category\}^+ &= \\ &\{name, category, color, dept, price\} \\ \{color\}^+ &= \{color\} \end{aligned}$$

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 entailed by 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^+

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

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

$F =$

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

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

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

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^+

$F =$

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

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

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

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

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

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

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^+

$F =$

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

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

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

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

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

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

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

Practice #3: Finding Closure1

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

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

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

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

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

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

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

Practice #3: Finding Closure2

■ Consider a schema with attributes $X=\{A,B,C,D,E,F,G,H\}$. In this practice, you are given a set of attributes $A\subset X$ and a set of FDs F .

- $A = \{A, B, F\}$
- $F = \{\{A, C\} \rightarrow \{D\},$
 $\{D, H, G\} \rightarrow \{E\},$
 $\{A, B\} \rightarrow \{G\},$
 $\{F, B, G\} \rightarrow \{C\}\}$

1) Compute A^+

2) Find one FD to add to F so that the closure $A^+=X$

3. Closures, Superkeys & Keys

What you will learn about in this section

1. Closures
2. Superkeys & Keys
3. **ACTIVITY:** The key or a key?

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 $A \in X^+$

Using Closure to Infer ALL FDs

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

Example:

Given $F =$

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

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

$\{B\} \rightarrow D$

$\{A\}^+ = \{A\}$

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

$\{C\}^+ = \{C\}$

$\{D\}^+ = \{D\}$

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

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

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

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

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

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

Using Closure to Infer ALL FDs

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

Example:

Given $F =$

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

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

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 :

Example:

Given $F =$

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

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

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

$$\begin{aligned} \{A\}^+ &= \{A\}, \{B\}^+ = \{B, D\}, \{C\}^+ = \{C\}, \{D\}^+ = \{D\}, \{A, B\}^+ = \{A, B, C, D\}, \\ \{A, C\}^+ &= \{A, C\}, \{A, D\}^+ = \{A, B, C, D\}, \{A, B, C\}^+ = \{A, B, D\}^+ = \{A, C, D\}^+ = \\ &\{A, B, C, D\}, \{B, C, D\}^+ = \{B, C, D\}, \{A, B, C, D\}^+ = \{A, B, C, D\} \end{aligned}$$

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

$$\begin{aligned} \{A, B\} &\rightarrow \{C, D\}, \{A, D\} \rightarrow \{B, C\}, \\ \{A, B, C\} &\rightarrow \{D\}, \{A, B, D\} \rightarrow \{C\}, \\ \{A, C, D\} &\rightarrow \{B\} \end{aligned}$$

“Y is in the closure of X”

Using Closure to Infer ALL FDs

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

$$\begin{aligned} \{A\}^+ &= \{A\}, \{B\}^+ = \{B, D\}, \{C\}^+ = \{C\}, \{D\}^+ = \{D\}, \{A, B\}^+ = \{A, B, C, D\}, \\ \{A, C\}^+ &= \{A, C\}, \{A, D\}^+ = \{A, B, C, D\}, \{A, B, C\}^+ = \{A, B, D\}^+ = \{A, C, D\}^+ = \\ &= \{A, B, C, D\}, \{B, C, D\}^+ = \{B, C, D\}, \{A, B, C, D\}^+ = \{A, B, C, D\} \end{aligned}$$

Example:

Given $F =$

$$\begin{aligned} \{A, B\} &\rightarrow C \\ \{A, D\} &\rightarrow B \\ \{B\} &\rightarrow D \end{aligned}$$

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

$$\begin{aligned} \{A, B\} &\rightarrow \{C, D\}, \{A, D\} \rightarrow \{B, C\}, \\ \{A, B, C\} &\rightarrow \{D\}, \{A, B, D\} \rightarrow \{C\}, \\ \{A, C, D\} &\rightarrow \{B\} \end{aligned}$$

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

Practice #4: Closure

1. Consider relation $R(A,B,C,D,E)$ with functional dependencies:
 - $AB \rightarrow C, C \rightarrow D, BD \rightarrow E$
 - Which of the following sets of attributes does not functionally determine E?
 - 1) AD
 - 2) ABC
 - 3) BC
 - 4) BCD

2. Let relation $R(A,B,C,D,E,F,G,H)$ satisfy the following functional dependencies:
 - $A \rightarrow B, CH \rightarrow A, B \rightarrow E, BD \rightarrow C, EG \rightarrow H, DE \rightarrow F$
 - Which of the following FDs is also guaranteed to be satisfied by R?
 - 1) $CDE \rightarrow AF$
 - 2) $BCD \rightarrow FH$
 - 3) $BED \rightarrow CF$
 - 4) $CGH \rightarrow BF$

3. Let relation $R(A,B,C,D,E)$ satisfy the following functional dependencies:

- $AB \rightarrow C, BC \rightarrow D, CD \rightarrow E, DE \rightarrow A, AE \rightarrow B$
- Which of the following FDs is also guaranteed to be satisfied by R ?
 - 1) $AD \rightarrow B$
 - 2) $CD \rightarrow B$
 - 3) $D \rightarrow B$
 - 4) $AD \rightarrow C$

4. Suppose relation $R(A,B,C)$ currently has only the tuple $(0,0,0)$, and it must always satisfy the functional dependencies $A \rightarrow B$ and $B \rightarrow C$. Which of the following tuples may be inserted into R legally?

- 1) $(2, 0, 1)$
- 2) $(0, 2, 1)$
- 3) $(1, 1, 0)$
- 4) $(0, 1, 0)$

Practice #5: Closure in SQL

■ Related to the Problem #4 in Practice #4

- Make a table R using your Oracle database
- Insert the tuple (0, 0, 0)
- For each tuple from 1) to 4), check given functional dependencies are satisfied or not using SQL
 - For each tuple, drop existing table and then make new table
 - Or you can make a table having new name (e.g., R1, R2, R3, and R4)

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

This means that no subset of a
key is also a superkey (i.e.,
dropping any attribute from the
key makes it no longer 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**

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} \rightarrow price
{category} \rightarrow color

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

= the set of all attributes

\Rightarrow this is a **superkey**

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

Practice #6: Finding superkeys

1. Consider relation $R(A,B,C,D,E)$ with functional dependencies:

- $D \rightarrow C, CE \rightarrow A, D \rightarrow A, AE \rightarrow D$
- Which of the following is a superkey?
 - 1) A
 - 2) BD
 - 3) CDE
 - 4) BDE

2. Consider relation $R(A,B,C,D,E,F)$ with functional dependencies:

- $CDE \rightarrow B, ACD \rightarrow F, BEF \rightarrow C, B \rightarrow D$
- Which of the following is a superkey?
 - 1) BDF
 - 2) ABE
 - 3) ADEF
 - 4) ABEF

3. Consider relation $R(A,B,C,D,E,F,G)$ with functional dependencies:

- $AB \rightarrow C$, $CD \rightarrow E$, $EF \rightarrow G$, $FG \rightarrow E$, $DE \rightarrow C$, and $BC \rightarrow A$
- Which of the following is a superkey?
 - 1) ABEF
 - 2) BDF
 - 3) BDEG
 - 4) BDFG

Practice #7: Finding super keys and keys

■ Given the schema $R=\{A,B,C\}$, define two FDs such that it has only two keys

- Remember the definition of super key: $\{A_1, \dots, A_n\} \rightarrow B$ where B is any other attribute (NOT set) except for $\{A_1, \dots, A_n\}$

■ Now, given the below relation R , define a set of FDs to result in the most keys possible.

- $R = \{A,B,C,D,E\}$
- How many keys can you make? Largest number wins it all!