DB Design Theory

Quick Recap

- Data Anomalies
- What is FD ?
- Defining and validating FD’s

What’s Next?

- “Good” vs. “Bad” FDs: Intuition
- Closures
- Minimal Cover

"Good" vs. "Bad" FDs

FDs are derived from the real-world constraints on the attributes.

- defined by Domain Expert.
- An FD must ***hold at all times***

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"*

- *No redundancy and anomalies*

“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*

Given a set of FDs (from user) our goal is to **eliminate the “Bad Ones".**

Finding Functional Dependencies

Example:

StudentGrade (rollNo, name, email, CourseID, grade)

Provided FDs:

1. *rollNo → name, email*
2. *email → rollNo*
3. *rollNo, CourseID → grade*

Given the provided FDs, which other FD's hold

{CourseID, email} → {grade}

Does this FD holds on all **instance…**

Which / how many other FDs do?!?

Finding Functional Dependencies

Equivalent to asking:

Given a set of FDs, $F = \{f_1, \ldots f_n\}$, does an FD **g** hold?

Inference problem: How do we decide?

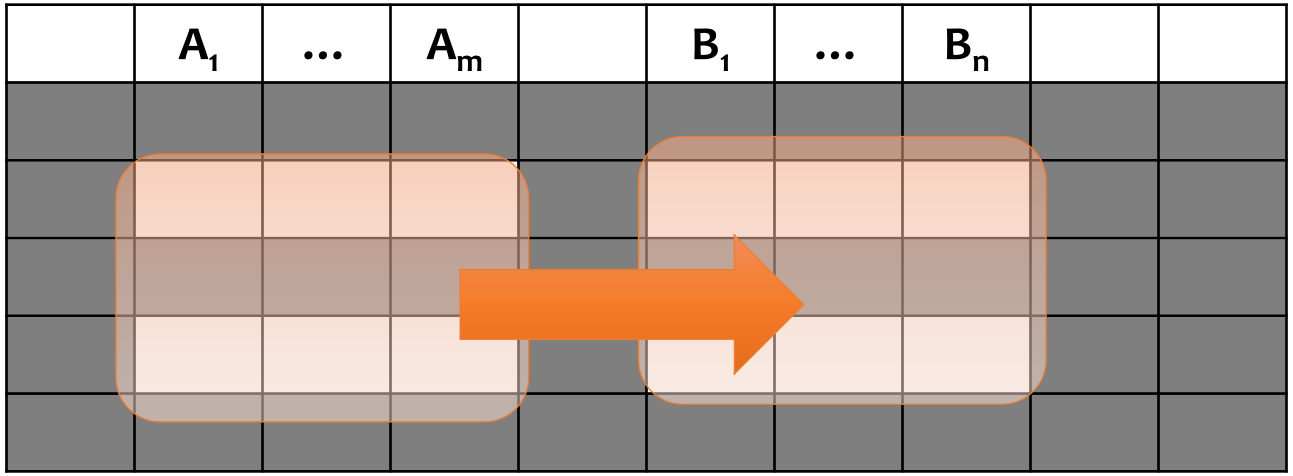
Three simple rules called **Armstrong's Rules.**

1. **Reduction (Reflexive)**
   - If $Y \subseteq X$, then X -> Y
2. **Augmentation**
   - If X -> Y, then XZ -> YZ
3. **Transitivity**
   - If X -> Y and Y -> Z, then X -> Z

Two further rules that can be derived from Armstrong's Rules

- **Union**
- **Decomposition**

Split (Decomposition)



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

… is equivalent to the following *n* FDs…

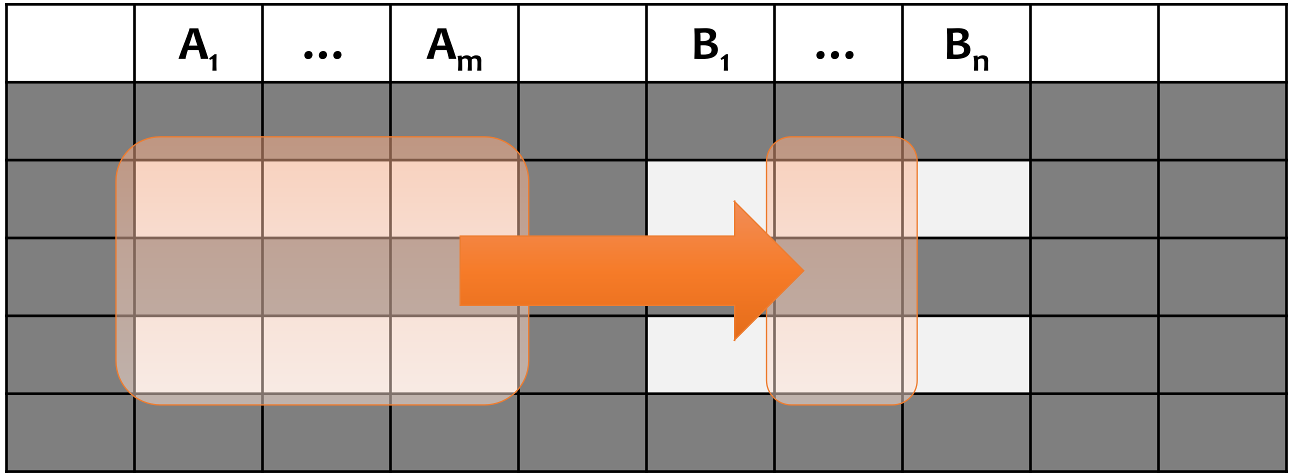
$A_1, \ldots, A_m \rightarrow B_1$

$A_1, \ldots, A_m \rightarrow B_2$

….

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

Combine (Union)



$A_1,\ldots,A_m \rightarrow B_1$
$A_1,\ldots,A_m \rightarrow B_2$
….
$A_1,\ldots,A_m \rightarrow B_i$ for $i=1,\ldots,n$

… is equivalent to …

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

Proofs

- If X -> YZ, then X -> Y and X -> Z
  - *YZ -> Y (reflexive)*
  - *X -> YZ (given)*
  - *X -> Y (transitivity)*
- If X -> Y and X -> Z, then X -> YZ
  - *X ->XY (augmenting X in X-> Y)*
  - *XY -> YZ (augmenting Y in X-> Z)*
  - *X -> YZ (transitivity)*
- If X -> Y and WY -> Z, then WX -> Z
  - *WX->WY (augmenting)*
  - *WY -> Z (given)*
  - *WX -> Z (transitivity)*

Armstrong's Rules

1. **Reduction (Reflexive)**
   - If Y $\subseteq$X, then X -> Y
2. **Augmentation**
   - If X -> Y, then XZ -> YZ
3. **Transitivity**
   - If X -> Y and Y -> Z, then X -> Z

Pseudo Transitivity

Finding Functional Dependencies

Example:

StudentGrade (rollNo, name, email, CourseID, grade)

Provided FDs:

1. *rollNo → name, email*
2. *email → rollNo*
3. *rollNo, CourseID → grade*

Given the provided FDs, which other FD's hold

Use Rules to determine which FD's hold ?

Using rules of FD's

- Given a relation *R and set of FD's F*
  - *Does another FD X ⟶ Y follow from F?*
  - *Use the rules to come up with a proof*

FDs

1. *rollNo ⟶ name, email*
2. *email ⟶ rollNo*
3. *rollNo, courseID ⟶ grade*

- Example: Which of the following FD holds
  1. ***email ⟶ name ?***
  2. ***name -> email ?***
  3. ***name , couseID -> name ?***

Using rules of FD's

- Given a relation *R and set of FD's F*
  - *Does another FD X → Y follow from F?*
  - *Use the rules to come up with a proof*

FDs

1. *rollNo → name, email*
2. *email → rollNo*
3. *rollNo, CourseID → grade*

- Example: ***courseID, email → grade?***
  - *email → rollNo (given in F)*
  - *courseID, email → courseID, rollNo (augmentation)*
  - *rollNo, courseID → grade (given in F)*
  - *courseID, email → grade (transitivity)*

Is there any algorithmic way to determine if an FD holds ?

Closures

Closure

Given a set of attributes $\mathbf{A_1, \dots, A_n}$ 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 =

1. *rollNo → name, email*
2. *email → rollNo*
3. *rollNo, CourseID → grade*

Example Closures:

$\{name\}^+$ =
$\{rollNo\}^+$ =

Activity: Closure Algorithm

StudentGrade (rollNo, name, email, courseID, grade)

- ***{ courseID, email }$^+$ = ?***
  - ***{ courseID, email }***$^+$ = *{courseID, email}*

  Consider FD: email → rollNo
  - ***{ courseID, email }***$^+$ = *{rollNo, courseID, email}*

  Consider FD: rollNo → name, email
  - ***{ courseID, email }***$^+$ = *{rollNo, courseID, email, name}*

  Consider FD: rollNo, courseID → grade
  - ***{ courseID, email }***$^+$ = *{rollNo, courseID, email, name, grade}*

FDs

1. *rollNo → name, email*
2. *email → rollNo*
3. *rollNo, courseID → grade*

Closure Algorithm

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

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

  if an FD $\{B_1, \ldots, B_n\} \rightarrow C$ is in F **and** $\{B_1, \ldots, B_n\} \subseteq X$

    then add C to **X**.

Return **X** as X^+

Closure 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**

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} → {Color}
2. {Category} → {Department}
3. {Color, Category} → {Price}

Example Closures:

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

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}

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

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}

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

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

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}

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} → {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}

Example

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

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

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

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

Example

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

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

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

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

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

Keys and Superkeys

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

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} → price
{category} → color

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

= the set of all attributes

⟹ this is a **superkey**

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

Algorithm: To Find a Key K for R given a set F of FD's

Input: A universal relation R and a set of functional dependencies F on the attributes of R.

1. Set $K := R$.
2. For each attribute A in K {

    compute $(K - A)^+$ with respect to F;

    If $(K - A)^+$ contains all the attributes in R,

        then set $K := K - \{A\}$;

    }

Activity: Find the Key

- Consider a relation R= {A,B,C,D,E,F,G,H,I,J} and
- The set of FD's
  - *A,B -> C*
  - *A->E*
  - *H->J*
  - *A,I-> D,H*
  - *B,I->G,F*

- Find the key for Relation R with given FD's.
- What if R2 = {A,B,C,D,E,F,G,H,I,J, **K**} and Fd's are same as above