CS 4604: Introduction to Database Management Systems

B. Aditya Prakash

Lecture #13: Functional Dependencies
Course Outline

- **Weeks 1–4: Query/Manipulation Languages and Data Modeling**
  - Relational Algebra
  - Data definition
  - Programming with SQL
  - Entity-Relationship (E/R) approach
  - Specifying Constraints
  - Good E/R design

- **Weeks 5–8: Indexes, Processing and Optimization**
  - Storing
  - Hashing/Sorting
  - Query Optimization
  - NoSQL and Hadoop

- **Week 9-10: Relational Design**
  - Functional Dependencies
  - Normalization to avoid redundancy

- **Week 11-12: Concurrency Control**
  - Transactions
  - Logging and Recovery

- **Week 13–14: Students’ choice**
  - Practice Problems
  - XML
  - Data mining and warehousing

Prakash 2018
Announcements

- Handout 3 on FDs and Normalization is out.
  - We will discuss it next Tue, Oct 23
Functional Dependencies and Schema Normalization

- A bit abstract and theoretical!
- But important!

Plan: 3 lectures
- 1. What are FDs? How to reason about them?
- 2. BCNF, 3NF and Normalization
- 3. Practice Problems in class
Overview

- Functional dependencies
  - why
  - Definition
  - Attribute closures and keys
  - Armstrong’s “axioms”
  - FD closure and cover
- Convert to relations
  - Students (ID, Name)
  - Advisors (ID, Name)
  - Favourite (StudentID, AdvisorID)
  - Advises (StudentID, AdvisorID)
What if we combine Students, Advises, and Favourite into one relation?

- Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)
- Seems ‘intuitively bad’ right?
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- What makes it bad?
- Given the Student’s Id, can any other values be determined?
  - Name and FavouriteAdvisorId
  - Id \(\rightarrow\) Name
  - Id \(\rightarrow\) FavouriteAdvisorId
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- Name and FavouriteAdvisorId
- Id $\rightarrow$ Name
- Id $\rightarrow$ FavouriteAdvisorId
- AdvisorId $\rightarrow$ ?
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)
  - Name and FavouriteAdvisorId
  - Id $\rightarrow$ Name
  - Id $\rightarrow$ FavouriteAdvisorId
  - AdvisorId $\rightarrow$ AdvisorName

- Can we say Id $\rightarrow$ AdvisorId?
  - Not really! Why?
  - Id is not a key for Students relation
  - Key: {Id, AdvisorId}
Example of a Bad Relation

Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- OK, what *really* makes it bad?
- Ans: Parts of the key determine other attributes
- Leads to:
  - Redundancy (Space, Inconsistencies, ....)
Motivation for Functional Dependencies

- Reason about constraints on attributes in a relation
- Procedurally determine the keys of a relation
- Detect when a relation has redundant information
- Improve database designs systematically using normalization
Overview

- Functional dependencies
  - why
  - Definition
  - Attribute closures and keys
  - Armstrong’s “axioms”
  - FD closure and cover
Definition of FD (Functional Dependency)

- $X \rightarrow Y$

‘X’ functionally determines ‘Y’

Informally: ‘if you know ‘X’, there is only one ‘Y’ to match’
Definition of FD (Functional Dependency)

- (If t is a tuple in a relation R and A is an attribute of R, then t[A] is the value of attribute A in tuple t)

- Formally:
  \[ X \rightarrow Y \rightarrow (t1[X] = t2[X] \rightarrow t1[Y] = t2[Y]) \]

if two tuples agree on the ‘X’ attribute, they *must* agree on the ‘Y’ attribute, too
(eg., if ids are the same, so should be names)
X \rightarrow Y

- X and Y can be sets of attributes

- Definition of FDs

- A FD on a relation R is a statement:
  - If two tuples in R agree on attributes A1, A2, ..., An they agree on attribute B
  - Notation: A1 A2 ... An \rightarrow B
A FD is a **constraint** on a single relational schema

- It must hold on every *instance* of the relation

- You cannot deduce an FD from a relation instance!

- (but you can deduce if an FD does NOT hold using an instance)
### Examples of FDs

- List the FDs

Courses(Number, DeptName, CourseName, Classroom, Enrollment)

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>Databases</td>
<td>TORG 1020</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>Dance</td>
<td>Tree Dancing</td>
<td>Drillfield</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>English</td>
<td>The Basis of Data</td>
<td>Williams 44</td>
<td>45</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>MCB 114</td>
<td>100</td>
</tr>
<tr>
<td>2604</td>
<td>Physics</td>
<td>Dark Matter</td>
<td>Williams 44</td>
<td>100</td>
</tr>
</tbody>
</table>

- Number DeptName → CourseName
- Number DeptName → Classroom
- Number DeptName → Enrollment
- Number DeptName → CourseName Classroom Enrollment
Examples of FDs

- List the FDs

Courses(Number, DeptName, CourseName, Classroom, Enrollment)

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>Databases</td>
<td>TORG 1020</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>Dance</td>
<td>Tree Dancing</td>
<td>Drillfield</td>
<td>45</td>
</tr>
<tr>
<td>4604</td>
<td>English</td>
<td>The Basis of Data</td>
<td>Williams 44</td>
<td>45</td>
</tr>
<tr>
<td>2604</td>
<td>CS</td>
<td>Data Structures</td>
<td>MCB 114</td>
<td>100</td>
</tr>
<tr>
<td>2604</td>
<td>Physics</td>
<td>Dark Matter</td>
<td>Williams 44</td>
<td>100</td>
</tr>
</tbody>
</table>

- Is Number → Enrollment an FD?
Where do FDs come from?

- “Keyness” of attributes
- Domain and application constraints
- Real world constraints
  - E.g. ProfessorID Time → Classroom
Definition of Keys

- FDs allow us to formally define keys
- A set of attributes \( \{A_1, A_2, \ldots, A_n\} \) is a key for relation \( R \) if:
  
  **Uniqueness:** \( \{A_1, A_2, \ldots, A_n\} \) functionally determine all the other attributes of \( R \)

  **Minimality:** no proper set of \( \{A_1, A_2, \ldots, A_n\} \) functionally determines all other attributes of \( R \).
Definitions of Keys

- A superkey is a set of attributes that has the uniqueness property but is not necessarily minimal
- If a relation has multiple keys, specify one to be primary key
- Convention: underline the attributes (but you know that!)
- If a key has only one attribute $A$, say $A$ rather than $\{A\}$
Example of keys

- What is the key for Courses (Number, DeptName, CourseName, Classroom, Enrollment) ?

- The key is \{Number, DeptName\}
  - These attributes functionally determine every other attribute
  - No proper subset of \{Number, DeptName\} has this property
Example of Keys

- What is the key for Teach (Number, DepartmentName, ProfessorName, Classroom) ?

- The key is \{Number, DepartmentName\}  
  – Why?
Keys in E/R to Relational Conversion

- From an ENTITY SET

If the relation comes from an entity set, the key attributes of the relation are precisely the key attributes of the entity set
Keys in E/R to Relational Conversion

- From a RELATIONSHIP (binary for now between E and F)
  - $R$ is many-many:
    - Key attributes of the relation are the key attributes of E and of F
  - $R$ is many-one:
    - Key attributes of the relation are the key attributes of E
  - $R$ is one-one:
    - Key attributes of the relation are the key attributes of E or of F
Keys in E/R to Relational Conversion

- From a RELATIONSHIP (multiway?)
- Need to reason about the FDs that R satisfies
- No simple rule
- If R has an arrow towards entity set E, at least one key for the relation for R excludes the key for E
Rules for Manipulating FDs

- Learn how to reason about FDs
- Define rules for deriving new FDs from a given set of FDs
- Example: R (A, B, C) satisfies FDs A \( \rightarrow \) B, B \( \rightarrow \) C.
  - What others does it satisfy?
  - A \( \rightarrow \) C
  - What is the key for R?
  - A (as A \( \rightarrow \) B and A \( \rightarrow \) C)
**Equivalence of FDs**

- **Why?**
  - To derive new FDs from a set of FDs
- **An FD F follows from a set of FDs T if**
  - every relation instance that satisfies all the FDs in T also satisfies F
  - $A \rightarrow C$ follows from $T = \{A \rightarrow B, B \rightarrow C\}$
- **Two sets of FDs S and T are equivalent if**
  - each FD in S follows from T and each FD in T follows from S
  - $S = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ and $T = \{A \rightarrow B, B \rightarrow C\}$ are equivalent
Splitting and Combining FDs

- The set of FDs
  - A1 A2 A3...An → B1
  - A1 A2 A3...An → B2
  - ...
  is equivalent to the FD
  - A1 A2 A3...An → B1 B2 B3 ... Bm

- This equivalence implies two rules:
  - Splitting rule
  - Combining rule
  - These rules work because all the FDs in S and T have identical left hand sides
Splitting and Combining FDs

- Can we split and combine left hand sides of FDs?
- For the relation Courses, is the FD
  - Number DeptName $\rightarrow$ CourseName equivalent to the set of FDs
    - $\{\text{Number } \rightarrow \text{CourseName}, \text{DeptName } \rightarrow \text{CourseName}\}$ ?
  - NO
Triviality of FDs

- A FD A1 A2...An → B1 B2...Bm is
  - Trivial if the B’s are a subset of the A’s
    \[ \{B_1, B_2, \ldots, B_n\} \subseteq \{A_1, A_2, \ldots, A_n\} \]
  - Non-trivial if at least one B is not among the A’s
    \[ \{B_1, B_2, \ldots, B_n\} - \{A_1, A_2, \ldots, A_n\} \neq \emptyset \]
  - Completely non-trivial if none of the B’s are among the A’s
    \[ \{B_1, B_2, \ldots, B_n\} \cap \{A_1, A_2, \ldots, A_n\} = \emptyset \]
Triviality of FDs

- What good are trivial and non-trivial FDs?
  - Trivial dependencies are always true
  - They help simplify reasoning about FDs

- Trivial dependency rule: The FD $A_1 \ A_2 \ldots \ A_n \rightarrow B_1 \ B_2 \ldots B_m$ is equivalent to the FD $A_1 \ A_2 \ldots \ A_n \rightarrow C_1 \ C_2 \ldots C_k$, where the $C$’s are those $B$’s that are not $A$’s i.e.

$$\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} - \{A_1, A_2, \ldots, A_n\}$$
Overview

- Functional dependencies
  - why
  - Definition
- Attribute closures and keys
  - Armstrong’s “axioms”
  - FD closure and cover
Closure of Attributes: Example

- Suppose a relation R (A, B, C, D, E, F) has FDs:
  - AB → C, BC → AD, D → E, CF → B

- Question:
  Find set X of attributes such that AB → X is true

- Answer:
  X = {A, B, C, D, E} i.e. AB → ABCDE
Closure of Attributes: Example

- Suppose a relation R (A, B, C, D, E, F) has FDs:
  - AB → C, BC → AD, D → E, CF → B

- Question:
  Find set Y of attributes such that BCF → Y is true

- Answer:
  Y = {A, B, C, D, E, F} i.e. BCF → ABCDEF

What is BCF?

ans: A superkey
Closure of Attributes: Example

- Suppose a relation R (A, B, C, D, E, F) has FDs:
  - AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B

- Question:
  Find set Z of attributes such that AF \rightarrow Z is true

- Answer:
  Y = \{A, F\} i.e. AF \rightarrow AF
Closure of Attributes: Example

- Suppose a relation R (A, B, C, D, E, F) has FDs:
  - AB → C, BC → AD, D → E, CF → B

- X, Y, Z are the closures of \{A, B\}, \{B, C, F\}, and \{A, F\} respectively
Attribute Closure, another way of looking (not in book)

R(A, B, C)

FD set:
AB->C (1)
A->BC (2)
B->C (3)
A->B (4)
**Closure of Attributes: Definition**

- **Given:**
  - Attributes \{A_1, A_2, \ldots, A_n\}
  - A set of FDs \(S\)

- **The **Closure** of \{A_1, A_2, \ldots, A_n\} under \(S\) is**
  - the set of attributes \{B_1, B_2, \ldots, B_m\} such that for \(1 \leq i \leq m\), the FD \(A_1 A_2 \ldots A_n \rightarrow B_i\) follows from \(S\)
  - the closure is denoted by \(\{A_1, A_2, \ldots, A_n\}^+\)
• Question:
Which attributes must \( \{A_1, A_2, \ldots, A_n\}^+ \) contain at the minimum?

• Answer:
\( \{A_1, A_2, \ldots, A_n\} \)

• Why?
A1 A2 ... An \( \rightarrow \) Ai is a trivial FD
Closure of Attributes: Algorithm

- **Given (INPUT):**
  - Attributes \{A_1, A_2, \ldots, A_n\}
  - Set of FDs \(S\)

- **Find (OUTPUT):**
  - \(X = \{A_1, A_2, \ldots, A_n\}^+\)
Closure of Attributes: Algorithm

1. Use the splitting rule so that each FD in S has one attribute on the right.
2. Set $X = \{A_1, A_2, ..., A_n\}$
3. Find FD $B_1 B_2 ... B_k \rightarrow C$ in S such that
   $\{B_1 B_2 ... B_k\} \subseteq X$ but $C \notin X$
4. Add $C$ to $X$
5. Repeat the last two steps until you can’t find $C$

Why is the algorithm correct?
Why compute Attribute Closures?

- Prove correctness of rules for manipulating FDs

Example:
Prove the transitive rule i.e.

If

\[ A_1 \ A_2 \ \ldots \ \ A_n \ \rightarrow \ B_1 \ B_2 \ \ldots \ \ B_m \]

\[ B_1 \ B_2 \ \ldots \ \ B_m \ \rightarrow \ C_1 \ C_2 \ \ldots \ \ C_k \]

Then

\[ A_1 \ A_2 \ \ldots \ \ A_n \ \rightarrow \ C_1 \ C_2 \ \ldots \ \ C_k \]

To prove this, check if

\[ \{C_1, C_2, \ldots, C_k\} \subseteq \{A_1, A_2, \ldots, A_n\}^+ \]
Why compute Attribute closures?

- Check if a “new” FD $A_1, A_2, \ldots, A_n \rightarrow B$ follows from a set of FDs $S$
- Simply check if $B$ is in $\{A_1, A_2, \ldots, A_n\}^+$ under $S$
- Get keys procedurally (aka algorithmically)

A set of attributes $X$ is a key for a relation $R$ iff
- $\{X\}^+$ is the set of all attributes of $R$
- For no attribute $A \in X$ is $\{X - \{A\}\}^+$ the set of all attributes of $R$
Examples of Closure Computations

- Consider the “bad” relation Students (Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)

- What are the FDs that hold in this relation?
  - Id \(\rightarrow\) Name
  - Id \(\rightarrow\) FavouriteAdvisorId
  - AdvisorId \(\rightarrow\) AdvisorName

- To compute the key for this relation:
  - Compute the closures for all sets of attributes
  - Find the minimal set of attributes whose closure is the set of all attributes
Algorithm for computing keys

- **Given (INPUT)**:
  - A relation $R (A_1, A_2, ..., A_n)$
  - The set of all FDs $S$ that hold in $R$

- **Find (OUTPUT)**:
  - Compute all the keys of $R$

1. For every subset $K$ of $\{A_1, A_2, ..., A_n\}$ compute its closure
2. If $\{K\}^+ = \{A_1, A_2, ..., A_n\}$ and for every attribute $A$, $\{K - \{A\}\}^+$ is not $\{A_1, A_2, ..., A_n\}$, then output $K$ as a key

- Running time?
Overview

- Functional dependencies
  - why
  - Definition
  - Attribute closures and keys
  - Armstrong’s “axioms”
  - FD closure and cover
Armstrong’s Axioms

- We can use closures of attributes to determine if any FD follows from a given set of FDs

OR

- Use Armstrong's axioms: complete set of inference rules from which it is possible to derive every FD that follows from a given set:
Armstrong’s Axioms

- **Reflexivity**
  \[ Y \subseteq X \implies X \implies Y \]
  - E.g. ssn, name \(\rightarrow\) ssn

- **Augmentation**
  \[ X \implies Y \implies XW \implies YW \]
  - E.g. ssn \(\rightarrow\) name then ssn grade \(\rightarrow\) name grade
Armstrong’s Axioms

- Transitivity

\[
\begin{align*}
X & \rightarrow Y \\
Y & \rightarrow Z
\end{align*}
\implies X \rightarrow Z
\]

e.g. if ssn \rightarrow address and address \rightarrow tax-rate

then

ssn \rightarrow tax-rate
Armstrong’s Axioms

Reflexivity: \[ Y \subseteq X \Rightarrow X \rightarrow Y \]

Augmentation: \[ X \rightarrow Y \Rightarrow XW \rightarrow YW \]

Transitivity: \[
\begin{align*}
X \rightarrow Y \\
Y \rightarrow Z
\end{align*}
\] \[ \Rightarrow X \rightarrow Z \]

‘sound’ and ‘complete’
Armstrong Axioms

- Additional rules
  - Union
    \[
    \begin{aligned}
    X \rightarrow Y \\
    X \rightarrow Z
    \end{aligned}
    \Rightarrow
    X \rightarrow YZ
    \]
  - Decomposition
    \[
    X \rightarrow YZ \Rightarrow
    \begin{aligned}
    X \rightarrow Y \\
    X \rightarrow Z
    \end{aligned}
    \]
  - Pseudo-transitivity
    \[
    \begin{aligned}
    X \rightarrow Y \\
    YW \rightarrow Z
    \end{aligned}
    \Rightarrow
    XW \rightarrow Z
    \]
Armstrong’s Axioms

- Prove ‘Union’ from three axioms:

\[ \begin{align*}
X \rightarrow Y \\
X \rightarrow Z
\end{align*} \implies X \rightarrow YZ \]
Armstrong’s Axioms

- Prove ‘Union’ from three axioms:

\[
egin{align*}
X & \rightarrow Y \quad (1) \\
X & \rightarrow Z \quad (2)
\end{align*}
\]

\[
(1) + \text{augm.} w / Z \Rightarrow XZ \rightarrow YZ \quad (3)
\]

\[
(2) + \text{augm.} w / X \Rightarrow XX \rightarrow XZ \quad (4)
\]

*but* \(XX\) *is* \(X\) *thus*

\[
(3) + (4) \quad \text{and transitivity} \quad \Rightarrow X \rightarrow YZ
\]
Armstrong’s Axioms

- Prove Pseudo-transitivity: try it

\[
\begin{align*}
Y \subseteq X &\implies X \rightarrow Y \\
X \rightarrow Y &\implies XW \rightarrow YW \\
X \rightarrow Y, Y \rightarrow Z &\implies X \rightarrow Z \\
X \rightarrow Y, YW \rightarrow Z &\implies XW \rightarrow Z
\end{align*}
\]
Armstrong’s Axioms

- Prove Decomposition: try it

\[ Y \subseteq X \implies X \rightarrow Y \]
\[ X \rightarrow Y \implies XW \rightarrow YW \]
\[ X \rightarrow Y, Y \rightarrow Z \implies X \rightarrow Z \]
\[ X \rightarrow YZ \implies \begin{cases} X \rightarrow Y \\ X \rightarrow Z \end{cases} \]
Relation Schema: $R(A_1, A_2, A_3)$: parentheses surround attributes, attributes separated by commas.

Set of attributes: $\{A_1, A_2, A_3\}$: curly braces surround attributes, attributes separated by commas.

FD: $A_1 A_2 \rightarrow A_3$: no parentheses or curly braces, attributes separated by spaces, arrows separates left hand side and right hand side.

Set of FDs: $\{A_1 A_2 \rightarrow A_3, A_2 \rightarrow A_1\}$: curly braces surround FDs, FDs separated by commas.
Overview

- Functional dependencies
  - why
  - Definition
  - Attribute closures and keys
  - Armstrong’s “axioms”
  - FD closure and cover
FDs - Closure $F^+$

Given a set $F$ of FD (on a schema)

$F^+$ is the set of all implied FD. Eg.,

takes($ssn$, $c$-$id$, grade, name, address)

  $ssn$, $c$-$id$ -> grade

  $ssn$-> name, address

$\{ \}$ $F$
FDs - Closure F+

ssn, c-id -> grade
ssn-> name, address
ssn-> ssn
ssn, c-id-> address
c-id, address-> c-id
...

F+
Computing Closures of FDs

- To compute the closure of a set of FDs, repeatedly apply Armstrong’s Axioms until you cannot find any new FDs.
Examples of Computing Closures of FDs

- (Let us include only completely non-trivial FDs in these examples, with a single attribute on the right)
- \( F = \{A \rightarrow B, B \rightarrow C\} \)
- \( \{F\}^+ = ?? \)
Examples of Computing Closures of FDs

- ((Let us include only completely non-trivial FDs in these examples, with a single attribute on the right))

- \( F = \{A \rightarrow B, B \rightarrow C\} \)

- \( \{F\}^+ = \{A \rightarrow B, B \rightarrow C, A \rightarrow C, AC \rightarrow B, AB \rightarrow C\} \)
Examples of Computing Closures of FDs

- ((Let us include only completely non-trivial FDs in these examples, with a single attribute on the right))
- \( F = \{AB \rightarrow C, BC \rightarrow A, AC \rightarrow B\} \)
- \( \{F\}^+ = ?? \)
Examples of Computing Closures of FDs

- ((Let us include only completely non-trivial FDs in these examples, with a single attribute on the right))
- $F = \{AB \rightarrow C, BC \rightarrow A, AC \rightarrow B\}$
- $\{F\}^+ = \{AB \rightarrow C, BC \rightarrow A, AC \rightarrow B\}$
Examples of Computing Closures of FDs

- ((Let us include only completely non-trivial FDs in these examples, with a single attribute on the right))
- \(F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\}\)
- \(\{F\}^+ = ??\)
Examples of Computing Closures of FDs

- ((Let us include only completely non-trivial FDs in these examples, with a single attribute on the right))

- \( F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D\} \)

- \( \{F\}^+ = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, A \rightarrow C, A \rightarrow D, B \rightarrow D, ...\} \)
Closures of Attributes vs Closure of FDs

- Both algorithms take as input a relation R and a set of FDs F
- Closure of FDs:
  - Computes \( \{F\}^+ \), the \textbf{set of all FDs} that follow from F
  - Output is a set of FDs
  - Output may contain an exponential number of FDs
- Closure of attributes:
  - In addition, takes a set \( \{A_1, A_2, \ldots, A_n\} \) of attributes as input
  - Computes \( \{A_1, A_2, \ldots, A_n\}^+ \), the \textbf{set of all attributes} B, such that \( A_1 A_2 \ldots A_n \rightarrow B \) follows from F
  - Output is set of all attributes
  - Output may contain at most the number of attributes in R
FDs - ‘canonical cover’ Fc

Given a set F of FD (on a schema) Fc is a **minimal set** of equivalent FDs. Eg.,
takes(ssn, c-id, grade, name, address)

- ssn, c-id -> grade
- ssn-> name, address
- ssn,name-> name, address
- ssn, c-id-> grade, name
Canonical cover

- Also sometimes called the ‘minimal basis’ or ‘minimal cover’
FDs - ‘canonical cover’ $Fc$

$Fc$

- $ssn, c\text{-id} \rightarrow \text{grade}$
- $ssn \rightarrow \text{name, address}$
- $ssn, \text{name} \rightarrow \text{name, address}$
- $ssn, c\text{-id} \rightarrow \text{grade, name}$
FDs - ‘canonical cover’ $F_c$

- why do we need it?
- define it properly
- compute it efficiently
FDs - ‘canonical cover’ Fc

- why do we need it?
  - easier to compute candidate keys
- define it properly
- compute it efficiently
FDs - ‘canonical cover’ $F_c$

- define it properly - three properties
  - 1) the RHS of every FD is a single attribute
  - 2) the closure of $F_c$ is identical to the closure of $F$ (ie., $F_c$ and $F$ are equivalent)
  - 3) $F_c$ is minimal (ie., if we eliminate any attribute from the LHS or RHS of a FD, property #2 is violated)
#3: we need to eliminate ‘extraneous’ attributes. An attribute is ‘extraneous if
– the closure is the same, before and after its elimination
– or if F-before implies F-after and vice-versa
FDs - ‘canonical cover’ Fc

\[\text{ssn, c-id} \rightarrow \text{grade} \]
\[\text{ssn} \rightarrow \text{name, address} \]
\[\text{ssn, name} \rightarrow \text{name, address} \]
\[\text{ssn, c-id} \rightarrow \text{grade, name} \]
FDs - ‘canonical cover’ Fc

Algorithm:
- examine each FD; drop extraneous LHS or RHS attributes; or redundant FDs
- make sure that FDs have a single attribute in their RHS
- repeat until no change
FDs - ‘canonical cover’ \( F_c \)

Trace algo for

\[
\begin{align*}
AB & \rightarrow C \quad (1) \\
A & \rightarrow BC \quad (2) \\
B & \rightarrow C \quad (3) \\
A & \rightarrow B \quad (4)
\end{align*}
\]
FDs - ‘canonical cover’ Fc

Trace algo for

AB->C (1)
A->BC (2)
B->C (3)
A->B (4)
split (2):

AB->C (1)
A->B (2')
A->C (2'')
B->C (3)
A->B (4)
FDs - 'canonical cover' $F_c$

AB$\rightarrow$C (1)
A$\rightarrow$B (2')
A$\rightarrow$C (2'')
B$\rightarrow$C (3)
A$\rightarrow$B (4)

AB$\rightarrow$C (1)
A$\rightarrow$C (2'')
B$\rightarrow$C (3)
A$\rightarrow$B (4)
FDs - ‘canonical cover’ Fc

\[
\begin{align*}
AB & \rightarrow C \quad (1) \\
A & \rightarrow C \quad (2') \quad \text{(redundant, implied by (4), (3), and transitivity)} \\
B & \rightarrow C \quad (3) \\
A & \rightarrow B \quad (4)
\end{align*}
\]
FDs - ‘canonical cover’ Fc

AB->C (1)
B->C (1’)
B->C (3)
B->C (3)
A->B (4)
A->B (4)

in (1), ‘A’ is extraneous:
(1),(3),(4) imply
(1’),(3),(4), and vice versa
FDs - ‘canonical cover’ Fc

- nothing is extraneous
- all RHS are single attributes
- final and original set of FDs are equivalent (same closure)
FDs - ‘canonical cover’ Fc

BEFORE

AB->C  (1)
A->BC  (2)
B->C   (3)
A->B   (4)

AFTER

B->C   (3)
A->B   (4)
Overview

- Functional dependencies
  - why
  - Definition
  - Attribute closures and keys
  - Armstrong’s “axioms”
  - FD closure and cover