FUNCTIONAL DEPENDENCIES
Functional Dependencies
Using FD’s to determine global IC’s:

Step 1: Given schema \( R = \{A_1, ..., A_n\} \)
use key constraints, laws of physics, trial-and-error, etc...
to determine an initial FD set, F.

Step 2: Use FD elimination techniques to generate an alternative (but equivalent) FD set, F’

Step 3: Write assertions for each f in F’. (for now)

Issues:
(1) How do we guarantee that \( F = F' \)?
ans: closures
(2) How do we find a “minimal” \( F' = F \)?
ans: minimal (canonical) cover algorithm
**Functional Dependencies**

Example:

Suppose \( R = \{ A, B, C, D, E, H \} \) and we determine that:

\[
F = \{ A \rightarrow BC, \\
B \rightarrow CE, \\
A \rightarrow E, \\
AC \rightarrow H, \\
D \rightarrow B \}
\]

Then we determine the minimal cover of \( F \):

\[
F_c = \{ A \rightarrow BH, \\
B \rightarrow CE, \\
D \rightarrow B \}
\]

ensuring that \( F \) and \( F_c \) are equivalent

Note: \( F \) requires 5 assertions

\( F_c \) requires 3 assertions
Functional Dependencies

Equivalence of FD sets:

FD sets F and G are equivalent if the imply the same set of FD’s

e.g. \( A \rightarrow B \) and \( B \rightarrow C \) : implies \( A \rightarrow C \)

equivalence usually expressed in terms of closures

Closures:
For any FD set, F, \( F^+ \) is the set of all FD’s implied by F.
can calculate in 2 ways:
   (1) Attribute Closure
   (2) Armstrong’s axioms

Both techniques tedious-- will do only for toy examples

F equivalent to G iff \( F^+ = G^+ \)
Given:

\[ R = \{ A, B, C, D, E, H\} \text{ and:} \]
\[ F = \{ A \rightarrow BC, \]
\[ B \rightarrow CE, \]
\[ A \rightarrow E, \]
\[ AC \rightarrow H, \]
\[ D \rightarrow B\} \]

What is the closure of CD \((CD^+)\)?

Algorithm att-closure \((X: \text{set of Attributes})\)

\[
\text{Result} \leftarrow X \\
\text{repeat until stable} \\
\text{for each } FD \text{ in } F, Y \rightarrow Z, \text{ do} \\
\text{if } Y \subseteq \text{Result} \text{ then} \\
\text{Result} \leftarrow \text{Result} \cup Z
\]
Q: what is $\text{ACD}^+$ ?
Ans: $\text{ACD}^+ \rightarrow R$

Q: How do you determine if ACD is a superkey?
Ans: it is if $\text{ACD}^+ \rightarrow R$

Q: How can you determine if ACD is a candidate key?
Ans: It is if:

$\text{ACD}^+ \rightarrow R$
$\text{AC}^+ \not\rightarrow R$
$\text{AD}^+ \not\rightarrow R$ not true => AD is a candidate key
$\text{CD}^+ \not\rightarrow R$
Attribute Closures to determine FD closures

Given:
\[ R = \{ A, B, C, D, E, H\} \] and:
\[ F = \{ A \rightarrow BC, \]
\[ B \rightarrow CE, \]
\[ A \rightarrow E, \]
\[ AC \rightarrow H, \]
\[ D \rightarrow B\} \]

\[ F^+ = \{ A \rightarrow A+, \]
\[ B \rightarrow B+, \]
\[ C \rightarrow C+, \]
\[ D \rightarrow D+, \]
\[ E \rightarrow E+, \]
\[ H \rightarrow H+, \]
\[ AB \rightarrow AB+, \]
\[ AC \rightarrow AC+, \]
\[ AD \rightarrow AD+, \]
\[ .......\} \]

To decide if F,G are equivalent:
(1) Compute \( F^+ \)
(2) Compute \( G^+ \)
(3) Is \( (1) = (2) \) ?

Expensive: \( F^+ \) has 63 rules (in general: \( 2^{|R|}-1 \) rules)
FD Closures Using Armstrong’s Axioms

A. Fundamental Rules (W, X, Y, Z: sets of attributes)
   1. Reflexivity
      If $Y \subseteq X$ then $X \rightarrow Y$
   2. Augmentation
      If $X \rightarrow Y$ then $WX \rightarrow WY$
   3. Transitivity
      If $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

B. Additional rules (can be proved from A)
   4. UNION: If $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$
   5. Decomposition: If $X \rightarrow YZ$ then $X \rightarrow Y$, $X \rightarrow Z$
   6. Pseudotransitivity: If $X \rightarrow Y$ and $WY \rightarrow Z$ then $WX \rightarrow Z$
FD Closures Using Armstrong’s Axioms

Given;

\[ F = \{ A \rightarrow BC, \quad (1) \]
\[ B \rightarrow CE, \quad (2) \]
\[ A \rightarrow E, \quad (3) \]
\[ AC \rightarrow H, \quad (4) \]
\[ D \rightarrow B \} \quad (5) \]

Exhaustively apply Armstrong’s axioms to generate \( F^+ \)

\[ F^+ = F \cup \]

1. \( \{ A \rightarrow B, A \rightarrow C \} \): decomposition on (1)
2. \( \{ A \rightarrow CE \} \): transitivity to 1.1 and (2)
3. \( \{ B \rightarrow C, B \rightarrow E \} \): decomp to (2)
4. \( \{ A \rightarrow C, A \rightarrow E \} \) decomp to 2
5. \( \{ A \rightarrow H \} \) pseudotransitivity to 1.2 and (4)
Our goal:
given a set of FD set, F, find an alternative FD set, G that is:
  smaller
  equivalent

Bad news:
Testing F=G (F+ = G+) is computationally expensive

Good news:
Minimal Cover (or Canonical Cover) algorithm:
given a set of FD, F, finds minimal FD set equivalent to F

Minimal: can’t find another equivalent FD set w/ fewer FD’s
Minimal Cover Algorithm

Given:

\[ F = \{ A \rightarrow BC, \]
\[ B \rightarrow CE, \]
\[ A \rightarrow E, \]
\[ AC \rightarrow H, \]
\[ D \rightarrow B \}\]

Determines minimal cover of F:

\[ Fc = \{ A \rightarrow BH, \]
\[ B \rightarrow CE, \]
\[ D \rightarrow B \}\]

• \( Fc = F \)
• No G that is equivalent to F and is smaller than \( Fc \)

Another example:

\[ F = \{ A \rightarrow BC, \]
\[ B \rightarrow C, \]
\[ A \rightarrow B, \]
\[ AB \rightarrow C, \]
\[ AC \rightarrow D \}\]

\[ \xrightarrow{MC\ \text{Algorithm}} \]

\[ Fc = \{ A \rightarrow BD, \]
\[ B \rightarrow C \}\]
Minimal Cover Algorithm

Basic Algorithm

ALGORITHM MinimalCover (X: FD set)
BEGIN
    REPEAT UNTIL STABLE
        (1) Where possible, apply UNION rule (A’s axioms)
            (e.g., A → BC, A → CD becomes A → BCD)
        (2) remove “extraneous attributes” from each FD
            (e.g., AB → C, A → B becomes
                 A → B, B → C
                 i.e., A is extraneous in AB → C)
Extraneous Attributes

(1) Extraneous is RHS?
   e.g.: can we replace $A \rightarrow BC$ with $A \rightarrow C$?
   (i.e. Is $B$ extraneous in $A \rightarrow BC$?)

(2) Extraneous in LHS?
   e.g.: can we replace $AB \rightarrow C$ with $A \rightarrow C$?
   (i.e. Is $B$ extraneous in $AB \rightarrow C$?)

Simple but expensive test:
   1. Replace $A \rightarrow BC$ (or $AB \rightarrow C$) with $A \rightarrow C$ in $F$
      
      $$F_2 = F - \{A \rightarrow BC\} \cup \{A \rightarrow C\}$$
      
      or
      
      $$F - \{AB \rightarrow C\} \cup \{A \rightarrow C\}$$

   2. Test if $F_{2+} = F_+ ?$
      if yes, then $B$ extraneous
A. RHS: Is B extraneous in A → BC?

step 1: F2 = F - \{A → BC\} U \{A → C\}
step 2: F+ = F2+ ?

To simplify step 2, observe that F2+ ⊆ F+

i.e., not new FD’s in F2+)

Why? Have effectively removed A→B from F

When is F+ = F2+ ?

Ans. When (A→B) in F2+

Idea: if F2+ includes: A→B and A→C, then it includes A→BC
Extraneous Attributes

B. LHS: Is B extraneous in A → B → C?

step 1: F2 = F - {AB → C} U {A → C}
step 2: F+ = F2+ ?

To simplify step 2, observe that F+ ⊆ F2+

i.e., there may be new FD’s in F2+)

Why? A → C “implies” AB → C. therefore all FD’s in F+ also in F2+.
But AB → C does not “imply” A → C

When is F+ = F2+?

Ans. When (A → C) in F+ Idea: if F+ includes: A → C then it will include all the FD’s of F+
A. RHS:
Given \( F = \{ A \rightarrow BC, B \rightarrow C \} \) is \( C \) extraneous in \( A \rightarrow BC \)?

why or why not?

Ans: yes, because

\[ A \rightarrow C \text{ in } \{ A \rightarrow B, B \rightarrow C \}^+ \]

Proof.
1. \( A \rightarrow B \)
2. \( B \rightarrow C \)
3. \( A \rightarrow C \) transitivity using Armstrong’s axioms
Extraneous attributes

B. LHS:
Given \( F = \{ A \rightarrow B, AB \rightarrow C \} \) is \( B \) extraneous in \( AB \rightarrow C \)?

why or why not?

Ans: yes, because

\[ A \rightarrow C \text{ in } F^+ \]

Proof.  
1. \( A \rightarrow B \)
2. \( AB \rightarrow C \)
3. \( A \rightarrow C \) using pseudotransitivity on 1 and 2

Actually, we have \( AA \rightarrow C \) but \( \{A, A\} = \{A\} \)
ALGORITHM MinimalCover (F: set of FD’s)
BEGIN
    REPEAT UNTIL STABLE
        (1) Where possible, apply UNION rule (A’s axioms)

        (2) Remove all extraneous attributes:
            a. Test if B extraneous in \( A \rightarrow BC \)
               (B extraneous if
                \( (A \rightarrow B) \) in \( (F - \{A \rightarrow BC\} U \{A \rightarrow C\})^+ \) )
            b. Test if B extraneous in \( AB \rightarrow C \)
               (B extraneous in \( AB \rightarrow C \) if
                \( (A \rightarrow C) \) in \( F^+ \) )
Minimal Cover Algorithm

Example: determine the minimal cover of

\[ F = \{A \rightarrow BC, \ B \rightarrow CE, \ A \rightarrow E\} \]

Iteration 1:

a. \[ F = \{A \rightarrow BCE, \ B \rightarrow CE\} \]

b. Must check for up to 5 extraneous attributes

- B extraneous in \( A \rightarrow BCE \)? No
- C extraneous in \( A \rightarrow BCE \)?
  
  yes: \( (A \rightarrow C) \) in \( \{A \rightarrow BE, \ B \rightarrow CE\} \)
  
  1. \( A \rightarrow BE \) -> 2. \( A \rightarrow B \) -> 3. \( A \rightarrow CE \) -> 4. \( A \rightarrow C \)

- E extraneous in \( A \rightarrow BE \)?
Minimal Cover Algorithm

Example: determine the minimal cover of
\[ F = \{A \rightarrow BC, B \rightarrow CE, A \rightarrow E\} \]

Iteration 1:
  a. \( F = \{ A \rightarrow BCE, B \rightarrow CE\} \)
  b. Must check for up to 5 extraneous attributes

- B extraneous in \( A \rightarrow BCE? \) No
- C extraneous in \( A \rightarrow BCE? \) Yes
- E extraneous in \( A \rightarrow BE? \)
  1. \( A \rightarrow B \rightarrow 2. A \rightarrow CE \rightarrow A \rightarrow E \)
- E extraneous in \( B \rightarrow CE \) No
- C extraneous in \( B \rightarrow CE \) No

Iteration 2:
  a. \( F = \{ A \rightarrow B, B \rightarrow CE\} \)
  b. Extraneous attributes:
     - C extraneous in \( B \rightarrow CE \) No
     - E extraneous in \( B \rightarrow CE \) No

DONE
Minimal Cover Algorithm

Find the minimal cover of

\[ F = \{ A \to BC, \]  
\[ B \to CE, \]  
\[ A \to E, \]  
\[ AC \to H, \]  
\[ D \to B \} \]

Ans: \[ F_c = \{ A \to BH, B \to CE, D \to B \} \]
Find two different minimal covers of:

\[ F = \{ A \rightarrow BC, \ B \rightarrow CA, \ C \rightarrow AB \} \]

Ans:

\[ Fc1 = \{ A \rightarrow B, \ B \rightarrow C, \ C \rightarrow A \} \]

and

\[ Fc2 = \{ A \rightarrow C, \ B \rightarrow A, \ C \rightarrow B \} \]
FD so far...

1. Minimal Cover algorithm
   • result (Fc) guaranteed to be the minimal FD set equivalent to F

2. Closure Algorithms
   a. Armstrong’s Axioms:
      more common use: test for extraneous attributes
      in C.C. algorithm
   b. Attribute closure:
      more common use: test for superkeys

3. Purposes
   a. minimize the cost of global integrity constraints
      so far: \( \text{min gic’ s } = |Fc| \)

      In fact.... Min gic’ s = 0
      (FD’ s for “normalization”)

   so far: \( \text{min gic’ s } = |Fc| \)
Another use of FD’s: Schema Design

Example:

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
<th>cname</th>
<th>lno</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Jones</td>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Johnson</td>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horse</td>
<td>1.7M</td>
<td>Jones</td>
<td>L-93</td>
<td>500</td>
</tr>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Hayes</td>
<td>L-17</td>
<td>1000</td>
</tr>
</tbody>
</table>

R: “Universal relation”

tuple meaning: Jones has a loan (L-17) for $1000 taken out at the Downtown branch in Bkln which has assets of $9M

Design:

+ : fast queries (no need for joins!)
- : redundancy:
    - update anomalies examples?
    - deletion anomalies
1. Decomposing the schema

\[ R = ( \text{bname}, \text{bcity}, \text{assets}, \text{cname}, \text{lno}, \text{amt}) \]

\[ R = R_1 \cup R_2 \]

\[ R_1 = (\text{bname}, \text{bcity}, \text{assets}, \text{cname}) \]

\[ R_1 = (\text{cname}, \text{lno}, \text{amt}) \]

2. Decomposing the instance

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
<th>cname</th>
<th>lno</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Jones</td>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Johnson</td>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horse</td>
<td>1.7M</td>
<td>Jones</td>
<td>L-93</td>
<td>500</td>
</tr>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Hayes</td>
<td>L-17</td>
<td>1000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cname</th>
<th>lno</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>L-17</td>
<td>1000</td>
</tr>
<tr>
<td>Johnson</td>
<td>L-23</td>
<td>2000</td>
</tr>
<tr>
<td>Jones</td>
<td>L-93</td>
<td>500</td>
</tr>
<tr>
<td>Hayes</td>
<td>L-17</td>
<td>1000</td>
</tr>
</tbody>
</table>
Goals of Decomposition

1. Lossless Joins
   Want to be able to reconstruct big (e.g. universal) relation by joining smaller ones (using natural joins)
   (i.e. \( R_1 \bowtie R_2 = R \))

2. Dependency preservation
   Want to minimize the cost of global integrity constraints based on FD’s
   (i.e. avoid big joins in assertions)

3. Redundancy Avoidance
   Avoid unnecessary data duplication (the motivation for decomposition)

Why important?
   LJ: information loss
   DP: efficiency (time)
   RA: efficiency (space), update anomalies
Dependency Goal #1: lossless joins

A bad decomposition:

<table>
<thead>
<tr>
<th>bname</th>
<th>bcity</th>
<th>assets</th>
<th>cname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Jones</td>
</tr>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Johnson</td>
</tr>
<tr>
<td>Mianus</td>
<td>Horse</td>
<td>1.7M</td>
<td>Jones</td>
</tr>
<tr>
<td>Downtown</td>
<td>Bkln</td>
<td>9M</td>
<td>Hayes</td>
</tr>
</tbody>
</table>

Problem: join adds meaningless tuples

“lossy join”: by adding noise, have lost meaningful information as a result of the decomposition
Is the following decomposition lossless or lossy?

<table>
<thead>
<tr>
<th>bname</th>
<th>assets</th>
<th>cname</th>
<th>lno</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>9M</td>
<td>Jones</td>
<td>L-17</td>
</tr>
<tr>
<td>Downtown</td>
<td>9M</td>
<td>Johnson</td>
<td>L-23</td>
</tr>
<tr>
<td>Mianus</td>
<td>1.7M</td>
<td>Jones</td>
<td>L-93</td>
</tr>
<tr>
<td>Downtown</td>
<td>9M</td>
<td>Hayes</td>
<td>L-17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>lno</th>
<th>bcity</th>
<th>amt</th>
</tr>
</thead>
<tbody>
<tr>
<td>L-17</td>
<td>Bkln</td>
<td>1000</td>
</tr>
<tr>
<td>L-23</td>
<td>Bkln</td>
<td>2000</td>
</tr>
<tr>
<td>L-93</td>
<td>Horse</td>
<td>500</td>
</tr>
</tbody>
</table>

Ans: Lossless: \( R = R_1 \Join R_2 \), it has 4 tuples
Ensuring Lossless Joins

A decomposition of $R : R = R_1 \cup R_2$ is lossless iff

$$R_1 \cap R_2 \rightarrow R_1,$$

or

$$R_1 \cap R_2 \rightarrow R_2$$

(i.e., intersecting attributes must be a superkey for one of the resulting smaller relations)
Decomposition Goal #2: Dependency preservation

Goal: efficient integrity checks of FD’s

An example w/ no DP:
R = ( bname, bcity, assets, cname, Ino, amt)
  bname → bcity  assets
  Ino → amt bname

Decomposition: R = R1 U R2
  R1 = (bname, assets, cname, Ino)
  R2 = (Ino, bcity, amt)

Lossless but not DP. Why?

Ans: bname → bcity assets  crosses 2 tables
Decomposition Goal #2: Dependency preservation

To ensure best possible efficiency of FD checks

ensure that only a SINGLE table is needed in order to check each FD

i.e. ensure that:  \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \)

Can be checked by examining \( R_i = (\ldots, A_1, A_2, \ldots, A_n, \ldots, B_1, \ldots, B_m, \ldots) \)

To test if the decomposition \( R = R_1 \cup R_2 \cup \ldots \cup R_n \) is DP

(1) see which FD’s of R are covered by \( R_1, R_2, \ldots, R_n \)

(2) compare the closure of (1) with the closure of FD’s of R
Example: Given $F = \{ A \rightarrow B, \ AB \rightarrow D, C \rightarrow D\}$

consider $R = R_1 \cup R_2$ s.t.
$R_1 = (A, B, C), \ R_2 = (C, D)$ is it DP?

(1) $F^+ = \{ A \rightarrow BD, \ C \rightarrow D\}^+$
(2) $G^+ = \{ A \rightarrow B, \ C \rightarrow D\}^+$

(3) $F^+ = G^+$? No because $(A \rightarrow D)$ not in $G^+$

Decomposition is not DP
Decomposition Goal #2: Dependency preservation

Example: Given $F = \{ A \rightarrow B, \ AB \rightarrow D, \ C \rightarrow D\}$

consider $R = R_1 \cup R_2$ s.t.

$R_1 = (A, B, D) \ , \ R_2 = (C, D)$

(1) $F^+ = \{ A \rightarrow BD, \ C \rightarrow D\}^+$

(2) $G^+ = \{A \rightarrow BD, \ C \rightarrow D, \ ...\}^+$

(3) $F^+ = G^+$

note: $G^+$ cannot introduce new FDs not in $F^+$

Decomposition is DP
Decomposition Goal #3: Redundancy Avoidance

Example:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>x</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>y</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>y</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>m</td>
<td>y</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>n</td>
<td>z</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>p</td>
<td>z</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Redundancy for B=x, y and z

(1) An FD that exists in the above relation is: B → C

(2) A superkey in the above relation is A, (or any set containing A)

When do you have redundancy?
   Ans: when there is some FD, X → Y covered by a relation and X is not a superkey
Normalization

Decomposition techniques for ensuring:
Lossless joins
Dependency preservation
Redundancy avoidance

We will look at some normal forms:
Boyce-Codd Normal Form (BCNF)
3rd Normal Form (3NF)