Design Theory for Relational Databases

csc343, Fall 2015
Diane Horton

Originally based on slides by Jeff Ullman
Introduction

◆ There are always many different schemas for a given set of data.

◆ E.g., you could combine or divide tables.

◆ How do you pick a schema? Which is better? What does “better” mean?

◆ Fortunately, there are some principles to guide us.
Database Design Theory

◆ It allows us to improve a schema systematically.
◆ General idea:
  ◦ Express constraints on the relationships between attributes
  ◦ Use these to decompose the relations
◆ Ultimately, get a schema that is in a “normal form” that guarantees good properties, such as no anomalies.
◆ “Normal” in the sense of conforming to a standard.
◆ The process of converting a schema to a normal form is called normalization.
Agenda

◆ Functional Dependencies (FD)
◆ Closure
◆ FD Projection
◆ Minimal Cover
◆ Normalization: BCNF, 3NF
Part I: Functional Dependency Theory
A poorly designed table

<table>
<thead>
<tr>
<th>part</th>
<th>manufacturer</th>
<th>manAddress</th>
<th>seller</th>
<th>sellerAddress</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>Hammers ‘R Us</td>
<td>99 Pinecrest</td>
<td>ABC</td>
<td>1229 Bloor W</td>
<td>5.59</td>
</tr>
<tr>
<td>8624</td>
<td>Lee Valley</td>
<td>102 Vaughn</td>
<td>ABC</td>
<td>1229 Bloor W</td>
<td>23.99</td>
</tr>
<tr>
<td>9141</td>
<td>Hammers ‘R Us</td>
<td>99 Pinecrest</td>
<td>ABC</td>
<td>1229 Bloor W</td>
<td>12.50</td>
</tr>
<tr>
<td>1983</td>
<td>Hammers ‘R Us</td>
<td>99 Pinecrest</td>
<td>Walmart</td>
<td>5289 St Clair W</td>
<td>4.99</td>
</tr>
</tbody>
</table>

◆ In any domain, there are relationships between attribute values.
◆ Perhaps:
  ◆ Every part has 1 manufacturer
  ◆ Every manufacture has 1 address
  ◆ Every seller has 1 address
◆ If so, this table will have redundant data.
Principle: Avoid redundancy

Redundant data can lead to anomalies.

<table>
<thead>
<tr>
<th>part</th>
<th>manufacturer</th>
<th>manAddress</th>
<th>seller</th>
<th>sellerAddress</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1983</td>
<td>Hammers ‘R Us</td>
<td>99 Pinecrest</td>
<td>ABC</td>
<td>1229 Bloor W</td>
<td>5.59</td>
</tr>
<tr>
<td>8624</td>
<td>Lee Valley</td>
<td>102 Vaughn</td>
<td>ABC</td>
<td>1229 Bloor W</td>
<td>23.99</td>
</tr>
<tr>
<td>9141</td>
<td>Hammers ‘R Us</td>
<td>99 Pinecrest</td>
<td>ABC</td>
<td>1229 Bloor W</td>
<td>12.50</td>
</tr>
<tr>
<td>1983</td>
<td>Hammers ‘R Us</td>
<td>99 Pinecrest</td>
<td>Walmart</td>
<td>5289 St Clair W</td>
<td>4.99</td>
</tr>
</tbody>
</table>

- **Update anomaly**: if Hammers ‘R Us moves and we update only one tuple, the data is inconsistent.
- **Deletion anomaly**: If ABC stops selling part 8624 and Lee Valley makes only that one part, we lose track of its address.
Definition of FD

◆ Suppose $R$ is a relation, and $X$ and $Y$ are subsets of the attributes of $R$.
◆ $X \rightarrow Y$ asserts that:
  ▶ If two tuples agree on all the attributes in set $X$, they must also agree on all the attributes in set $Y$.
◆ We say that “$X \rightarrow Y$ holds in $R$”, or “$X$ functionally determines $Y$.”
◆ An FD constrains what can go in a relation.
A → B means:

∀ tuples \( t_1, t_2, \)
(\( t_1[A] = t_2[A] \)) \( \Rightarrow \) (\( t_1[B] = t_2[B] \))

Or equivalently:

\( \neg \exists \) tuples \( t_1, t_2 \) such that
(\( t_1[A] = t_2[A] \)) \( \land \) (\( t_1[B] \neq t_2[B] \))
Generalization to multiple attributes

$A_1A_2 \ldots A_m \rightarrow B_1B_2 \ldots B_n$ means:

∀ tuples $t_1, t_2,$

$(t_1[A_1] = t_2[A_1] \land \ldots \land t_1[A_m] = t_2[A_m]) \Rightarrow$

$(t_1[B_1] = t_2[B_1] \land \ldots \land t_1[B_n] = t_2[B_n])$

Or equivalently:

$\neg \exists$ tuples $t_1, t_2$ such that

$(t_1[A_1] = t_2[A_1] \land \ldots \land t_1[A_m] = t_2[A_m]) \land$

$\neg (t_1[B_1] = t_2[B_1] \land \ldots \land t_1[B_n] = t_2[B_n])$
Why “functional dependency”? 

◆ “dependency” because the value of \( Y \) depends on the value of \( X \).

◆ “functional” because there is a function that takes a value for \( X \) and gives a unique value for \( Y \).

◆ (It’s not a typical function; just a lookup.)
Equivalent sets of FDs

◆ When we write a set of FDs, we mean that all of them hold.
◆ We can very often rewrite sets of FDs in equivalent ways.
◆ When we say $S_1$ is equivalent to $S_2$ we mean that:
  ◦ $S_1$ holds in a relation iff $S_2$ does.
FD - Exercises

🔹 1. Create an instance of R that violates BC → D
   ◦ Any tuple with equal values of B, C but unequal D values!

🔹 2.a) Is the FD A→BC equivalent to the two FDs A→B, A→C?
   ◦ Yes!

🔹 2.b) Is the FD PQ→R equivalent to the two FDs P→Q, P→R?
   ◦ No.
   ◦ Example..
     P | Q | R
     --------------
     2 | 1 | 4
     2 | 3 | 5
Splitting rules for FDs

◆ Can we split the RHS of an FD and get multiple, equivalent FDs?

◆ Can we split the LHS of an FD and get multiple, equivalent FDs?
Coincidence or FD?

- An FD is an assertion about every instance of the relation.
- You can’t know it holds just by looking at one instance.
- You must use knowledge of the domain to determine whether an FD holds.
FDs are closely related to keys

◆ Suppose K is a set of attributes for relation R.

◆ Our old definition of superkey:
    a set of attributes for which no two rows can have the same values.

◆ A claim about FDs:
    \( K \) is a superkey for \( R \) iff
    \( K \) functionally determines all of \( R \).
FDs are a *generalization* of keys

- **Superkey:**
  \[ X \rightarrow R \]
  All attributes

- **Functional dependency:**
  \[ X \rightarrow Y \]

- A superkey must include *all* the attributes of the relation on the RHS.
- An FD can have just a subset of them.
Inferring FDs

◆ Given a set of FDs, we can often infer further FDs.
◆ This will come in handy when we apply FDs to the problem of database design.
◆ Big task: given a set of FDs, infer every other FD that must also hold.
◆ Simpler task: given a set of FDs, infer whether a given FD must also hold.
Examples

◆ If $A \rightarrow B$ and $B \rightarrow C$ hold, must $A \rightarrow C$ hold?

◆ If $A \rightarrow H$, $C \rightarrow F$, and $FG \rightarrow AD$ hold, must $FA \rightarrow D$ hold? must $CG \rightarrow FH$ hold?

◆ Aside: we are not generating new FDs, but testing specific possible FD(s).
Prove an FD (LHS->RHS) **follows**
Method 1: using first principles

◆ You can prove it by referring back to
  ◦ The FDs that you know hold, and
  ◦ Apply FD inference rules (axioms)

◆ But the **Closure Test** is easier!
Prove an FD (LHS->RHS) follows
Method 2: using the Closure Test

◆ Assume you know the values of the LHS attributes, and figure out:
   everything else that is determined.

<e.g. restaurant name> ➔ everything I can learn about it...?

◆ If the result includes the RHS attributes, then you know that LHS -> RHS holds

◆ This is called the closure test.
**Y** is a set of attributes, **S** is a set of **FDs**.
Return the closure of **Y** under **S**.

**Attribute\_closure**(\(Y, S\)):

**Initialize** \(Y^+\) **to** \(Y\)

Repeat until no more changes occur:

If there is an FD \(LHS \rightarrow RHS\) in \(S\) such that LHS is in \(Y^+\):
Add RHS to \(Y^+\)

**Return** \(Y^+\)

If LHS is in \(Y^+\) and LHS \(\rightarrow\) RHS holds, we can add RHS to \(Y^+\)
Visualizing attribute closure

If LHS is in $Y^+$ and LHS $\rightarrow$ RHS holds, we can add RHS to $Y^+$.
\textit{S} is a set of FDs; \textit{LHS} \rightarrow \textit{RHS} is a single FD. Return true iff \textit{LHS} \rightarrow \textit{RHS} follows from \textit{S}.

\textbf{FD\_follows(S, LHS \rightarrow RHS):}

\begin{align*}
Y^+ &= \text{Attribute\_closure}(\text{LHS}, S) \\
\text{return} &\quad (\text{RHS is in } Y^+) 
\end{align*}
Exercise

Suppose we have a relation on attributes ABCDEF, with FDs:
\[ AC \rightarrow F, \ CEF \rightarrow B, \ C \rightarrow D, \ DC \rightarrow A \]

a) Does it follow that \( C \rightarrow F \)?

Ans: \( C^+ = \ldots? \)
\( C^+ = CDAF \)
Then, \( C \rightarrow F \) follows

b) Does it follow that \( ACD \rightarrow B \)?

Ans: \( ACD^+ = \ldots? \)
\( ACD^+ = ACDF \)
Then, \( ACD \rightarrow B \) does not follow
Projecting FDs

Later, we will learn how to **normalize** a schema by **decomposing** relations. (This is the whole point of this theory.)

We will need to be aware of what FDs hold in the new, smaller, relations.

In other words, we must **project our FDs** onto the attributes of our new relations.
Exercise - Projecting FDs

Suppose we have a relation on attributes **ABCDE** with FDs:  
\[ A \rightarrow C, \quad C \rightarrow E, \quad E \rightarrow BD \]

**Project the FDs onto attributes **ABC**:  

To project onto a set of attributes, we systematically consider every possible LHS of an FD that might hold on those attributes.

\[
\begin{align*}
A^+ &= \\
B^+ &= \\
C^+ &= \\
AB^+ &= \\
AC^+ &= \\
BC^+ &= 
\end{align*}
\]
Exercise - Projecting FDs

- Suppose we have a relation on attributes **ABCDE** with FDs:  
  \[ A \rightarrow C, \quad C \rightarrow E, \quad E \rightarrow BD \]

**Project the FDs onto attributes ABC:**

- To project onto a set of attributes, we systematically consider every possible LHS of an FD that might hold on those attributes.

  \[ A^+ = \{ \text{ACEBD} \} \]
  therefore \( A \rightarrow BC \).

  \[ B^+ = \{ B \} \]
  Yields no FDs for our set of attributes.

  \[ C^+ = \{ \text{CEBD} \} \]
  therefore \( C \rightarrow B \).

- We don’t need to consider any supersets of \( A \). \( A \) already determines all of our attributes ABC, so supersets of \( A \) will be only yield FDs that already follow from \( A \rightarrow BC \).

- The only superset left is \( BC \):

  \[ BC^+ = \{ \text{BCED} \} \]
  This yields no FDs for our set of attributes.

- So the projection of the FDs onto ABC is: \{A→BC, C→B \}
$S$ is a set of FDs; $L$ is a set of attributes.

Return the projection of $S$ onto $L$:

all FDs that follow from $S$ and involve only attributes from $L$.

**Project($S$, $L$):**

Initialize $T$ to $\emptyset$.

For each subset $X$ of $L$:

Compute $X^+$ _Close $X$ and see what we get._

For every attribute $A$ in $X^+$:

If $A$ is in $L$: _$X \rightarrow A$ is only relevant if $A$ is in $L$ (we know $X$ is)._ 

add $X \rightarrow A$ to $T$.

Return $T$. 
A few speed-ups

✿ No need to add $X \rightarrow A$ if $A$ is in $X$ itself. It’s a trivial FD.

✿ These subsets of $X$ won’t yield anything, so no need to compute their closures:
  - the empty set
  - the set of all attributes

✿ Neither are big savings, but ...
A big speed-up

If we find $X^+ = \text{all attributes}$, we can ignore any *superset* of $X$.

- It can only give use “weaker” FDs (with more on the LHS).

This is a big time saver!
Projection is expensive

- Even with these speed-ups, projection is still expensive.

- Suppose $R_1$ has $n$ attributes. How many subsets of $R_1$ are there?
  
  ($A$ set of $n$ elements has $2^n$ subsets)
Minimal Basis (aka Minimal Cover)

- We saw earlier that we can very often rewrite sets of FDs in equivalent ways.
- Example: $S_1 = \{A -> BC\}$ is equivalent to $S_2 = \{A -> B, A -> C\}$.
- Given a set of FDs $S$, we may want to find a **minimal basis**: A set of FDs that is equivalent, but has
  1. No redundant FDs
  2. No unnecessary attributes on the LHS
S is a set of FDs. Return a minimal basis for S.

**Minimal\_basis(S):**

Repeat until no more changes result:

1. Remove FDs that are implied by the rest.
2. For each FD with $2^+$ attributes on the left:
   If you can remove one attribute from the LHS and get an FD that follows from the rest:
   Do so! (It’s a stronger FD.)
Example – Minimal Basis

What is the minimal cover of these FDs in ABCDEG:

\[ A \rightarrow B, \quad ABCD \rightarrow E, \quad G \rightarrow A, \quad G \rightarrow B \]

Answer:

(1) Can any of the FDs be implied from another FD? (if so, drop)

*Systematic approach:

- Calculate the closure of the LHS in each FD, using the rest of FDs; Can we reach the RHS using the other FDs?

1. \( A \rightarrow B \)
   
   \( A^+ \) under \{2,3,4\} \? = ...

2. \( ABCD \rightarrow E \)
   
   \( ABCD^+ \) under \{1,3,4\} \? = ...

3. \( G \rightarrow A \)
   
   \( G^+ \) under \{1,2,4\} \? = ...

4. \( G \rightarrow B \)
   
   \( G^+ \) under \{1,2,3\} \? = GAB

- Drop FD#4.
Example – Minimal Basis

◆ What is the minimal cover of these FDs in ABCDEG:

\[ A \rightarrow B, \quad ABCD \rightarrow E, \quad G \rightarrow A, \quad G \rightarrow B \]

Answer:

(2) Check if any LHS can be reduced (any attributes can be removed?). If so, drop the extra attributes.

1. \( A \rightarrow B \)

2. \( ABCD \rightarrow E \)

   \textit{Start with removing 1 attribute..} (ACD+, ABC+, BCD+, ..and so on)

   \( ACD+ = ABCD \)

3. \( G \rightarrow A \)
Example – Minimal Basis

What is the minimal cover of these FDs in ABCDEG:

\[ A \rightarrow B, \quad ABCD \rightarrow E, \quad G \rightarrow A, \quad G \rightarrow B \]

Answer:

(2) Check if any LHS can be reduced (any attributes can be removed?). If so, drop the extra attributes.

1. A \rightarrow B
2. ABCD \rightarrow E
   
   \textit{Start with removing 1 attribute..} (ACD+, ABC+, BCD+, ..and so on)
   
   ACD+ = ABCD

2. ACD \rightarrow E
3. G \rightarrow A

Result: Minimal basis is \ A \rightarrow B, \ ACD \rightarrow E, \ G \rightarrow A

*Please check course website for more detailed examples*
Some comments on computing a minimal basis

◆ Often there are multiple possible results, depending on the order in which you consider the possible simplifications.

◆ After you identify a **redundant** FD, you must **not** use it when computing any subsequent closures (as you consider whether other FDs are redundant).
... and some that are less intuitive

- When you are computing closures to decide whether the LHS of an FD
  \[ a_1a_2...a_m \rightarrow b_1b_2...b_n \]
  can be simplified, continue to use that FD.

- When you have tried to eliminate each FD and to reduce each LHS, you must go back and try again.