Design Theory for Relational Databases

csc343, Fall 2015
Bogdan Simion
University of Toronto

Originally based on slides by
Diane Horton & 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.
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 manufacturer 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.
◆ [Exercise]
A -> B means:

∀ tuples \( t_1, t_2, \)

\[(t_1[A] = t_2[A]) \Rightarrow (t_1[B] = t_2[B])\]

Or equivalently:

¬∃ tuples \( t_1, t_2 \) such that

\[(t_1[A] = t_2[A]) \land (t_1[B] ≠ t_2[B])\]
Generalization to multiple attributes

\( A_1A_2 \ldots A_m \rightarrow B_1B_2 \ldots B_n \) means:

\[ \forall \text{tuples } t_1, t_2, \]
\[ (t_1[A_1] = t_2[A_1] \land \ldots \land t_1[A_m] = t_2[A_m]) \Rightarrow \]
\[ (t_1[B_1] = t_2[B_1] \land \ldots \land t_1[B_n] = t_2[B_n]) \]

Or equivalently:

\[ \neg \exists \text{tuples } t_1, t_2 \text{ such that} \]
\[ (t_1[A_1] = t_2[A_1] \land \ldots \land t_1[A_m] = t_2[A_m]) \land \]
\[ \neg (t_1[B_1] = t_2[B_1] \land \ldots \land t_1[B_n] = t_2[B_n]) \]
Why “functional dependency”? 

◆“dependency” because the value of $Y$ depends on the value of $X$.
◆“functional” because there is a mathematical 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.
Splitting rules for FDs

◆ Can we split the RHS of an FD and get multiple, equivalent FDs?
  ♦ A->BC splits into: A->B, A->C ?

◆ Can we split the LHS of an FD and get multiple, equivalent FDs?
  ♦ AB->C splits into: A->C, B->C ?

◆ [Exercise]
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?
  - AB->C splits into: A->C, B->C?

- [Exercise]
Rules for functional dependencies

- **Combining rule**
  - $A \rightarrow B_1, A \rightarrow B_2, \ldots A \rightarrow B_n$
  - $\Rightarrow A \rightarrow B_1, B_2, \ldots B_n$

- **Trivial dependency (S1->S2, where S2 $\subseteq$ S1)**
  - Assume S1 and S1 are sets of attributes
  - $S_1 \rightarrow S_2 \Rightarrow S_1 \rightarrow S_1 \cup S_2$
  - $S_1 \rightarrow S_2 \Rightarrow S_1 \rightarrow S_1 \cap S_2$
Armstrong’s axioms

◆ Reflexivity: \( AB \subseteq ABC \implies ABC \rightarrow AB \)
◆ Augmentation: \( A \rightarrow B \implies AC \rightarrow BC \)
◆ Transitivity: \( A \rightarrow B, B \rightarrow C \implies A \rightarrow C \)

◆ Plus: Inference rules
  ♦ Union: \( X \rightarrow Y, X \rightarrow Z \implies X \rightarrow YZ \)
  ♦ Decomposition: \( X \rightarrow YZ \implies X \rightarrow Y, X \rightarrow Z \)
  ♦ Pseudo-transitivity: \( X \rightarrow Y, WY \rightarrow Z \implies WX \rightarrow Z \)
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:
  \[ X \rightarrow \{R\} \]
  Every attribute

◆ Functional dependency:
  \[ X \rightarrow Y \]

◆ A superkey: must include \textit{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 \text{ and } B \rightarrow C \text{ hold, must } A \rightarrow C \text{ hold?}

◆ If A \rightarrow H, C \rightarrow F, \text{ and } F G \rightarrow A D \text{ hold, must } F A \rightarrow D \text{ hold? must } C G \rightarrow F H \text{ hold?}

◆ If H \rightarrow GD, HD \rightarrow CE, \text{ and } BD \rightarrow A \text{ hold, must } EH \rightarrow C \text{ hold?}

◆ Aside: we are not generating new FDs, but testing a specific possible one.
Method 1: Prove an FD follows using first principles

- You can prove it by referring back to
  - The FDs that you know hold
  - The definition of functional dependency
- Apply rules to prove it
Method 1: Prove that an FD follows

\[ \text{R(ABCDEHI)} \]
\[ \text{FDs: \{A->B, A->C, CG->H, CG->I, B->H\}} \]

\[ \text{Are these also valid FDs? Why / why not?} \]

\[ \text{A->H} \]
  - Yes, using transitivity!

\[ \text{CG->HI} \]
  - Yes, using union rule!

\[ \text{AG->I} \]
  - A->C => AG -> CG => AG -> I
  - Or directly, using pseudo-transitivity
Another example

\[ R = \{ABCDEF\} \]

\[ \text{FDs: \{a. } AB \rightarrow E, \ b. \ BE \rightarrow F, \ c. \ E \rightarrow C, \ d. \ CF \rightarrow D\} \]

\[ \text{Prove that } AB \rightarrow CD \text{ follows.} \]

- Step1. \( AB \rightarrow E \) (given: a) and \( AB \rightarrow B \) (reflexivity)  
  \( \Rightarrow AB \rightarrow BE \) (union)

- Step2. \( AB \rightarrow F \) (transitivity: Step1 and b)

- Step3: \( AB \rightarrow C \) (transitivity: a & c)

- Step4: \( AB \rightarrow CF \) (union: Steps 2 & 3)

- Step5: \( AB \rightarrow D \) (transitivity: Step4 & d)

- Step6: \( AB \rightarrow CD \) (union: Step3 & 5)

\[ \text{That's fun and all, but might take too long to infer..} \]

\[ \text{The Closure Test is way easier!} \]
Method 2: Prove an FD follows using the Closure Test

- Assume you know the values of the LHS attributes, and figure out everything else that is determined (closure).
- If it includes the RHS attributes, then you know that LHS $\rightarrow$ RHS
- 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 -> RHS in S such that LHS is in Y^+:
         Add RHS to Y^+
   Return Y^+
Visualizing attribute closure

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

\[
\text{FD\_follows}(S, \text{LHS} \rightarrow \text{RHS}): \\
Y^+ = \text{Attribute\_closure}(\text{LHS}, S) \\
\text{return (RHS is in } Y^+) \\
\]

[Exercise]
Recall previous example

◆ R = {ABCDEF}

◆ FDs: {a. AB -> E, b. BE -> F, c. E -> C, d. CF -> D}

◆ Prove that AB -> CD follows. Use closure test now.

- Only one step: Calculate AB+
  - AB+ = AB
  - AB+ = ABE (from a)
  - AB+ = ABECF (from b and c)
  - AB+ = ABECFD (from d)
  - CD is in the closure of AB
  
  => AB -> CD holds.

◆ Much simpler, right?

TOO EASY
Exercise 5:
Prove that the sets \{A->BC\} and \{A->B, A->C\} are equivalent, using closures.

This can be proven using the closure test, as follows:

Assume that A -> BC.
- Under this assumption, A+ = ABC.
- Since B and C are in the closure of A => A -> B, and A -> C.

Assume that A -> B, and A -> C.
- Under this assumption, A+ = ABC.
- Therefore, since BC is in the closure of A => A -> BC.

Therefore, each set of FDs follows from the other. They are equivalent.
More closure examples

◆ R(ABCDE)

◆ FDs: A→D, B→C, D→B, E→B

◆ Determine A+
  • Start with \( A^+ = A \)
  • A→D \( \Rightarrow \) \( A^+ = AD \)
  • D→B \( \Rightarrow \) \( A^+ = ABD \)
  • B→C \( \Rightarrow \) \( A^+ = ABCD \)

◆ Is A a superkey?
  • No, can’t get to E
More closure examples

◆ R(ABCDE)

◆ FDs: A->D, B->C, D->B, E->B

◆ Determine D+
  ◆ Start with D+ = D
  ◆ D->B => D+ = BD
  ◆ B->C => D+ = BCD

◆ Determine E+
  ◆ Start with E+ = E
  ◆ E->B => E+ = BE
  ◆ B->C => E+ = BCE
More closure examples

◆ R(ABCDE)
◆ FDs: A->D, B->C, D->B, E->B
◆ Determine BD+
  ♦ Start with BD+ = BD
  ♦ D->B  =>  nothing changes
  ♦ B->C  =>  BD+ = BCD
◆ Determine AE+
  ♦ Start with AE+ = AE
  ♦ A->D, D->B, B->C  =>  AE+ = ABCDE
  ♦ AE -> R  =>  AE is a superkey!
  ♦ Is this the only key?
Today

- Projecting FDs
- Calculating minimal bases
Big picture

◆ 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.
Example

\( R(A_1, \ldots, A_n) \) Set of attributes: \( A \)

Decompose into:
- \( R_1(B_1, \ldots, B_k) \) Set of attributes: \( B \), and
- \( R_2(C_1, \ldots, C_m) \) Set of attributes: \( C \)

\( B \cup C = A, \quad R_1 \Join R_2 = R \)

\( R_1 = \pi_B(R) \)

\( R_2 = \pi_C(R) \)
Projecting FDs

- R1 = $\pi_B(R)$, R2 = $\pi_c(R)$
- Then what FDs hold in each of R1 and R2?
  - Project FDs!
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 {}.

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.
Exercise - Projecting FDs

Suppose we have a relation on attributes ABCDE with FDs: \( A \rightarrow C, \ C \rightarrow E, \ E \rightarrow BD \)

Project the FDs onto attributes ABC:

- A) Calculate closures for every subset \( X \) of ABC
- B) Look in each closure and infer non-trivial FDs (\( X \) on the LHS and a subset of the attributes ABC on the RHS).

\[
A^+ = \\
B^+ = \\
C^+ = \\
AB^+ = \\
AC^+ = \\
BC^+ =
\]
Exercise - Projecting FDs

Suppose we have a relation on attributes ABCDE with FDs: \( A \rightarrow C, \ C \rightarrow E, \ E \rightarrow BD \)

**Project the FDs onto attributes ABC:**

- A) Calculate closures for every subset \( X \) of ABC
- B) Look in each closure and infer non-trivial FDs (\( X \) on the LHS and a subset of the attributes ABC on the RHS).

\[ A^+ = \text{ACEBD} \]

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 \( BC \): \( BC^+ = BCED \). This yields no FDs for our set.

- So the projection of the FDs onto ABC is: \( \{ A \rightarrow BC, \ C \rightarrow B \} \)
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 us “weaker” FDs (with more on the LHS).

This is a big time saver!

Exercises 1b and 2
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?
  
  \[ (A \text{ set of } n \text{ elements has } 2^n \text{ subsets}) \]

- Keep in mind big picture: we ultimately want a minimal set of completely non-trivial FDs, such that all FDs that hold on the relation follow from the dependencies in this minimal set.
Minimal Basis (or Minimal Cover)

- We saw earlier that we can very often rewrite sets of FDs in equivalent ways.
- Example 1:
  - $S_1 = \{A \rightarrow BC\}$ is equivalent to $S_2 = \{A \rightarrow B, A \rightarrow C\}$.
- Example 2: $S_1 = \{A \rightarrow B, AB \rightarrow C\}$
  - $A^+ = ABC \implies A \rightarrow C$ also holds
  - So, we can eliminate the B from the 2nd FD
- Example 3: $S_1 = \{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$
  - We can get the 3rd by transitivity, so we don’t really need it.
Minimal Bases

- Given a set of FDs $S$, we may want to find a **minimal basis**: A set of FDs that is equivalent, but has
  - singleton RHS, and
  - no FDs with unnecessary attributes on the LHS, and
  - no redundant FDs
- These are basically the steps for our algorithm!
- You should do the steps in this order, otherwise you may get incorrect results! There’s a way to reverse them but it’s a bit more complex.
\(S\) is a set of FDs. Return a minimal basis for \(S\).

Minimal\_basis(S):

1. Break up non-singleton RHS into all FDs
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.)
3. Remove FDs that are implied by the rest.

[Exercise]
Example – Minimal Basis

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

\[ A \rightarrow B, \quad BC \rightarrow E, \quad C \rightarrow D, \quad D \rightarrow AE \]

* Systematic approach:

1) Singletons in RHS: \( A \rightarrow B, \quad BC \rightarrow E, \quad C \rightarrow D, \quad D \rightarrow A, D \rightarrow E \)
2) No FDs with unnecessary attributes on the LHS – if so, simplify:

\[ BC \rightarrow E \quad B^+ = B \implies \text{no help} \]

\[ C^+ = CDABE \implies \text{Contains B, so } C \rightarrow E \text{ is sufficient} \]

3) Can any of the FDs be implied from another FD? If so, drop:

We now have:
\[ A \rightarrow B, \quad C \rightarrow E, \quad C \rightarrow D, \quad D \rightarrow A, D \rightarrow E \]

Let’s take each one and see if it’s redundant:

\[ A \rightarrow B: \quad \text{Ignore it and calculate } A^+. \text{ See if it contains B. If so, drop this FD.} \]

\[ A^+ = A \implies \text{it’s a keeper!} \]

\[ C \rightarrow E: \quad C^+ = CDABE \implies \text{Redundant, because we got to E another way!} \]

\[ C \rightarrow D: \quad C^+ = C \text{ (remember: ignoring } C \rightarrow D \text{ and } C \rightarrow E \text{ is gone)} \implies \text{Keep!} \]

\[ D \rightarrow A: \quad D^+ = DE \implies \text{Keep!} \]

\[ D \rightarrow E: \quad D^+ = DAB \implies \text{Keep!} \]

Minimal cover:
\[ A \rightarrow B, \quad C \rightarrow D, \quad D \rightarrow A, D \rightarrow E \]
Exercise

Find a minimal basis for:

\[ S = \{\text{ABF} \rightarrow \text{G}; \ \text{BC} \rightarrow \text{H}; \ \text{BCH} \rightarrow \text{EG}; \ \text{BE} \rightarrow \text{GH}\} \]
Some comments on computing a minimal basis

◆ There may be 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).
Next time

◆ Normal forms