FUNCTIONAL DEPENDENCIES

CS 564- Spring 2018

ACKs: Dan Suciu, Jignesh Patel, AnHai Doan
WHAT IS THIS LECTURE ABOUT?

Database Design Theory:

• Functional Dependencies
• Armstrong’s rules
• The Closure Algorithm
• Keys and Superkeys
HOW TO BUILD A DB APPLICATION

• Pick an application
• Figure out what to model (ER model)
  – Output: ER diagram
• Transform the ER diagram to a relational schema

• Refine the relational schema (normalization)

• Now ready to implement the schema and load the data!
DB DESIGN THEORY

- Helps us identify the “bad” schemas and improve them
  1. express constraints on the data: functional dependencies (FDs)
  2. use the FDs to decompose the relations

- The process, called normalization, obtains a schema in a “normal form” that guarantees certain properties
  - examples of normal forms: BCNF, 3NF, ...

CS 564 [Spring 2018] - Paris Koutris
MOTIVATING EXAMPLE

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

- What is the primary key?
  - (SSN, PhoneNumber)
- What is the problem with this schema?
# MOTIVATING EXAMPLE

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

## Problems:
- redundant storage
- **update**: change the age of Paris?
- **insert**: what if a person has no phone number?
- **delete**: what if Arun deletes his phone number?
### SOLUTION: DECOMPOSITION

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>
FUNCTIONAL DEPENDENCIES
Functional dependencies (FDs) are a form of constraint. They generalize the concept of keys.

If two tuples agree on the attributes

\[ A = A_1, A_2, ..., A_n \]

then they must agree on the attributes

\[ B = B_1, B_2, ..., B_m \]

Formally:

\[ A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_m \]

We then say that \( A \) functionally determines \( B \).
# FD: EXAMPLE 1

<table>
<thead>
<tr>
<th>SSN</th>
<th>name</th>
<th>age</th>
<th>phoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>608-374-8422</td>
</tr>
<tr>
<td>934729837</td>
<td>Paris</td>
<td>24</td>
<td>603-534-8399</td>
</tr>
<tr>
<td>123123645</td>
<td>John</td>
<td>30</td>
<td>608-321-1163</td>
</tr>
<tr>
<td>384475687</td>
<td>Arun</td>
<td>20</td>
<td>206-473-8221</td>
</tr>
</tbody>
</table>

- $SSN \rightarrow name, age$
- $SSN, age \rightarrow name$
### FD: EXAMPLE 2

<table>
<thead>
<tr>
<th>studentID</th>
<th>semester</th>
<th>courseNo</th>
<th>section</th>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>124434</td>
<td>4</td>
<td>CS 564</td>
<td>1</td>
<td>Paris</td>
</tr>
<tr>
<td>546364</td>
<td>4</td>
<td>CS 564</td>
<td>2</td>
<td>Arun</td>
</tr>
<tr>
<td>999492</td>
<td>6</td>
<td>CS 764</td>
<td>1</td>
<td>Anhai</td>
</tr>
<tr>
<td>183349</td>
<td>6</td>
<td>CS 784</td>
<td>1</td>
<td>Jeff</td>
</tr>
</tbody>
</table>

- **courseNo, section → instructor**
- **studentID → semester**
SPLITTING AN FD

- Consider the FD: $A, B \rightarrow C, D$

- The attributes on the right are independently determined by $A, B$ so we can split the FD into:
  - $A, B \rightarrow C$ and $A, B \rightarrow D$

- We can not do the same with attributes on the left!
  - writing $A \rightarrow C, D$ and $B \rightarrow C, D$ does not express the same constraint!
TRIVIAL FDS

- Not all FDs are informative:
  - $A \rightarrow A$ holds for any relation
  - $A, B, C \rightarrow C$ also holds for any relation

- An FD $X \rightarrow A$ is called **trivial** if the attribute $A$ belongs in the attribute set $X$
  - a trivial FD always holds!
HOW TO IDENTIFY FDS

• An FD is domain knowledge:
  – an inherent property of the application & data
  – not something we can infer from a set of tuples

• Given a table with a set of tuples
  – we can confirm that a FD seems to be valid
  – to infer that a FD is definitely invalid
  – we can never prove that a FD is valid


**EXAMPLE 3**

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Tweaker</td>
<td>Gadget</td>
<td>Black</td>
<td>Toys</td>
<td>99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Stationary</td>
<td>Green</td>
<td>Office-supplies</td>
<td>59</td>
</tr>
</tbody>
</table>

**Q1:** Is $\text{name} \rightarrow \text{department}$ an FD?
- not possible!

**Q2:** Is $\text{name, category} \rightarrow \text{department}$ an FD?
- we don’t know!
WHY FDS?

1. keys are special cases of FDs
2. more integrity constraints for the application
3. having FDs will help us detect that a schema has redundancies and tell us how to normalize it
MORE ON FDS

• If the following FDs hold:
  – \( A \rightarrow B \)
  – \( B \rightarrow C \)

then the following FD is also true:
  – \( A \rightarrow C \)

• We can find more FDs like that using what we call Armstrong’s Axioms
ARMSTRONG’S AXIOMS: 1

Reflexivity
For any subset $X \subseteq \{A_1, \ldots, A_n\}$:
$$A_1, A_2, ..., A_n \rightarrow X$$

• Examples
  – $A, B \rightarrow B$
  – $A, B, C \rightarrow A, B$
  – $A, B, C \rightarrow A, B, C$
**Armstrong’s Axioms: 2**

**Augmentation**
For any attribute sets $X$, $Y$, $Z$:
if $X \rightarrow Y$ then $X, Z \rightarrow Y, Z$

- **Examples**
  - $A \rightarrow B$ implies $A, C \rightarrow B, C$
  - $A, B \rightarrow C$ implies $A, B, C \rightarrow C$
ARMSTRONG’S AXIOMS: 3

Transitivity
For any attribute sets $X, Y, Z$:
  if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

• Examples
  – $A \rightarrow B$ and $B \rightarrow C$ imply $A \rightarrow C$
  – $A \rightarrow C, D$ and $C, D \rightarrow E$ imply $A \rightarrow E$
APPLYING ARMSTRONG’S AXIOMS

Product(name, category, color, department, price)

1. name → color
2. category → department
3. color, category → price

• Infer: name, category → price
  1. We apply the augmentation axiom to (1) to obtain (4) name, category → color, category
  2. We apply the transitivity axiom to (4), (3) to obtain name, category → price
**APPLYING ARMSTRONG’S AXIOMS**

**Product** (name, category, color, department, price)

1. $name \rightarrow color$
2. $category \rightarrow department$
3. $color, category \rightarrow price$

- **Infer: $name, category \rightarrow color$**
  1. We apply the **reflexivity** axiom to obtain (5) $name, category \rightarrow name$
  2. We apply the **transitivity** axiom to (5), (1) to obtain $name, category \rightarrow color$
FD CLOSURE

FD Closure
If $F$ is a set of FDs, the closure $F^+$ is the set of all FDs logically implied by $F$

Armstrong’s axioms are:

- **sound**: any FD generated by an axiom belongs in $F^+$
- **complete**: repeated application of the axioms will generate all FDs in $F^+$
CLOSURE OF ATTRIBUTE SETS

Attribute Closure
If $X$ is an attribute set, the closure $X^+$ is the set of all attributes $B$ such that:

$$X \rightarrow B$$

In other words, $X^+$ includes all attributes that are functionally determined from $X$
EXAMPLE

Product(name, category, color, department, price)
• $name \rightarrow color$
• $category \rightarrow department$
• $color, category \rightarrow price$

Attribute Closure:
• $\{name\}^+ = \{name, color\}$
• $\{name, category\}^+ = \{name, color, category, department, price\}$
THE CLOSURE ALGORITHM

• Let $X = \{A_1, A_2, \ldots, A_n\}$
• **UNTIL** $X$ doesn’t change **REPEAT**:
  
  **IF** $B_1, B_2, \ldots, B_m \rightarrow C$ is an FD **AND**
  
  $B_1, B_2, \ldots, B_m$ are all in $X$

  **THEN** add $C$ to $X$
EXAMPLE

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

- \( A, B \rightarrow C \)
- \( A, D \rightarrow E \)
- \( B \rightarrow D \)
- \( A, F \rightarrow B \)

Compute the attribute closures:

- \( \{A, B\}^+ = \{A, B, C, D, E\} \)
- \( \{A, F\}^+ = \{A, F, B, D, E, C\} \)
WHY IS CLOSURE NEEDED?

1. Does $X \rightarrow Y$ hold?
   - we can check if $Y \subseteq X^+$

2. To compute the closure $F^+$ of FDs
   - for each subset of attributes $X$, compute $X^+$
   - for each subset of attributes $Y \subseteq X^+$, output the FD $X \rightarrow Y$
**Keys & Superkeys**

**Superkey**: a set of attributes $A_1, A_2, \ldots, A_n$ such that for any other attribute $B$ in the relation:

$$A_1, A_2, \ldots, A_n \rightarrow B$$

**Key** (or candidate key): a *minimal* superkey

– none of its subsets functionally determines all attributes of the relation

If a relation has multiple keys, we specify one to be the **primary key**
COMPUTING KEYS & SUPERKEYS

- Compute $X^+$ for all sets of attributes $X$
- If $X^+ = \text{all attributes}$, then $X$ is a superkey
- If no subset of $X$ is a superkey, then $X$ is also a key
EXAMPLE

Product\((\text{name, category, price, color})\)
• \(\text{name} \rightarrow \text{color}\)
• \(\text{color, category} \rightarrow \text{price}\)

Superkeys:
• \(\{\text{name, category}\}, \{\text{name, category, price}\}\)
  \(\{\text{name, category, color}\}, \{\text{name, category, price, color}\}\)

Keys:
• \(\{\text{name, category}\}\)
HOW MANY KEYS?

Q: Is it possible to have many keys in a relation $R$?

YES!! Take relation $R(A, B, C)$ with FDs

- $A, B \rightarrow C$
- $A, C \rightarrow B$
MINIMAL BASIS FOR FDS

• Given a set $F$ of FDs, we know how to compute the closure $F^+$
• A minimal basis of $F$ is the opposite of closure
• $S$ is a **minimal basis** for a set $F$ if FDs if:
  – $S^+ = F^+$
  – every FD in $S$ has one attribute on the right side
  – if we remove any FD from $S$, the closure is not $F^+$
  – if for any FD in $S$ we remove one or more attributes from the left side, the closure is not $F^+$
Example:

- $A \rightarrow B$
- $A, B, C, D \rightarrow E$
- $E, F \rightarrow G, H$
- $A, C, D, F \rightarrow E, G$
STEP 1: SPLIT THE RIGHT HAND SIDE

- $A \rightarrow B$
- $A, B, C, D \rightarrow E$
- $E, F \rightarrow G$
- $E, F \rightarrow H$
- $A, C, D, F \rightarrow E$
- $A, C, D, F \rightarrow G$
STEP 2: REMOVE REDUNDANT FDS

- $A \rightarrow B$
- $A, B, C, D \rightarrow E$
- $E, F \rightarrow G$
- $E, F \rightarrow H$
- $A, C, D, F \rightarrow E$
- $A, C, D, F \rightarrow G$

can be removed, since these FDs are logically implied by the remaining FDs
STEP 3: CLEAN UP THE LEFT HAND SIDE

- $A \rightarrow B$
- $A, B, C, D \rightarrow E$
- $E, F \rightarrow G$
- $E, F \rightarrow H$

B can be safely removed because of the first FD
EXAMPLE: FINAL RESULT

- $A \rightarrow B$
- $A, C, D \rightarrow E$
- $E, F \rightarrow G$
- $E, F \rightarrow H$
RECAP

- FDs and (super)keys
- Reasoning with FDs:
  - given a set of FDs, infer all implied FDs
  - given a set of attributes $X$, infer all attributes that are functionally determined by $X$
- Next we will look at how to use them to detect that a table is “bad”