Today

- Onto 3NF

Assignments

- HW 8, Proj 1 due
- HW 9, PROJ 2 out, due Tues
- Exam 2 next Thurs
Dependency-Preserving Decompositions

Decompositions should also *preserve dependencies*

- For example:

  Employee(eid, address, city, state, zip)
  
  - zip → state  
  - address, city, state → zip  

  (note not totally accurate)

- A possible decomposition

  Employee(eid, address, city, zip)

  ZipState(zip, state)

- While in BCNF, it does not preserve the FD:

  - address, city, state → zip

- In the example, we could wind up with this

  Employee  | ZipState
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>eid</td>
<td>address</td>
<td>city</td>
<td>zip</td>
<td>zip</td>
<td>state</td>
<td></td>
<td></td>
</tr>
<tr>
<td>123</td>
<td>111 1st Ave</td>
<td>Spokane</td>
<td>99202</td>
<td>99202</td>
<td>WA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>456</td>
<td>111 1st Ave</td>
<td>Spokane</td>
<td>99258</td>
<td>99258</td>
<td>WA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

  We’ve lost the ability to enforce the FD: address, city, state → zip
Dependency Preserving Decompositions

Defining “dependency preserving”

- Let \( F \) be the FDs of a relation \( R \)
- Let \( A \) and \( B \) be sets of attributes in \( R \)

- An FD \( X \rightarrow Y \) is “in \( A \)” if all attributes of \( X \) and \( Y \) are in \( A \)

- The “projection \( F_A \)” of \( F \) on attributes \( A \) are the FDs in \( A \)

- Assume \( R \) is decomposed into sets of attributes \( A \) and \( B \)
- The decomposition is “dependency preserving” if \((F_A \cup F_B)^+ = F^+\)
  
  – Essentially, can we get \( F \) back from just \( F_A \) and \( F_B \)

Example

- \( F = \{\text{address, city, state } \rightarrow \text{zip, zip } \rightarrow \text{state}\} \)

Q: If \( A = \{\text{eid, address, city, zip}\} \) and \( B = \{\text{zip, state}\} \), what are \( F_A \) and \( F_B \)?
  
  – \( F_A = \emptyset \) neither FD is in \( A \)
  
  – \( F_B = \{\text{zip } \rightarrow \text{state}\} \)

- In this case, \((F_A \cup F_B)^+ \) cannot equal \( F^+ \) since \( F_B^+ \neq F^+ \)
- And so this decomposition is not dependency preserving!
Third Normal Form (3NF)

Some schemas do not have both:

- a lossless decomposition
- and a dependency preserving decomposition into BCNF schemas

But, every schema has a lossless dep.-preserving decomposition into 3NF

A schema \( R \) is in 3NF if for every FD \( X \rightarrow Y \), either:

- \( X \rightarrow Y \) is a trivial FD (\( Y \subseteq X \)) from BCNF
- \( X \rightarrow Y \) is a key FD (\( X \) is a superkey) from BCNF
- \( Y \) is a part of some key for \( R \) not allowed in BCNF

Thus 3NF subsumes BCNF (every BCNF schema is also in 3NF)

Q: How could you decompose Employee into 3NF?

Employee(eid, address, city, state, zip)

- \( F = \{ \text{address, city, state} \rightarrow \text{zip}, \text{zip} \rightarrow \text{state} \} \)

- Use the FD address, city, state \( \rightarrow \) zip to obtain:

  Employee(eid, address, city, state)
  Location(address, city, state, zip)

  - This is a lossless decomposition Why?
  - This is a dependency preserving decomposition Why?
  - But it is in 3NF because of the FD zip \( \rightarrow \) state
  - Is zip \( \rightarrow \) state enforced by the DBMS? No!
Canonical Covers and 3NF

Consider this example

\[ R(a, b, c, d, e) \]

\[ F = \{ ab \rightarrow cde, a \rightarrow c, b \rightarrow d, d \rightarrow e \} \]

Q: What additional dependencies to we get when computing \( F^+ \)?

- The main one: \( b \rightarrow e \) (transitivity)

Q: Is there a non dependency-preserving decomposition into BCNF?

- Yes ...
- Step 1: Using \( a \rightarrow c \)
  \[ R(a, b, d, e), R1(a, c) \]
- Step 2: Using \( b \rightarrow d \)
  \[ R(a, b, e), R1(a, c), R2(b, d) \]
- Step 3: Using \( b \rightarrow e \)
  \[ R(a, b), R1(a, c), R2(b, d), R3(b, e) \]
- Note that \( (F_R \cup F_{R1} \cup F_{R2} \cup F_{R3})^+ \) does not contain \( d \rightarrow e \)
- Note also that other decompositions into BCNF are dependency preserving
  \( a \rightarrow c, d \rightarrow e \), then \( b \rightarrow d \)
Let's create a 3NF decomposition for this example ...

We first have to compute a canonical cover $F_c$ of $F$

- A set of dependencies “logically equivalent” to $F$ such that:
  - no functional dependency in $F_c$ contains an “extraneous” attribute
  - each left side of an FD is unique

**Extraneous attributes**

- if we can remove the attribute from a dependency in $F$
- without changing the closure of $F$

**A cover for the example**

- Start with:
  - $ab \rightarrow cde$
  - $a \rightarrow c$
  - $b \rightarrow d$
  - $d \rightarrow e$
  - $b \rightarrow e$

- Combine common left-hand sides:
  - $ab \rightarrow cde$
  - $a \rightarrow c$
  - $b \rightarrow de$
  - $d \rightarrow e$

- Remove extraneous attributes
- ab \rightarrow \emptyset \\
- a \rightarrow c \\
- b \rightarrow d \\
- d \rightarrow e \\

(a \rightarrow c, b \rightarrow d e)