Lecture 16:
• Normalization (wrap up)
• ER Diagrams (intro)

Announcements:
• Project out (multiple due dates)
• HW-5 out, due Thur, Nov 3
• PS-3 due Tues, Nov 8

Properties of Decompositions

A “good” decomposition is considered:

(1) **Lossless** ... can get the original table back
   • i.e., we haven’t lost information by decomposing into smaller tables

(2) **Dependency Preserving** ... can still enforce all of the FDs
   • i.e., we haven’t lost any of the functional dependencies we want to enforce
   • note in the previous example, order FDs applied mattered
   • algorithms exist that don’t rely on FD order

(3) In BCNF (if possible) or else 3NF
Checking Decomposition Properties

Lossless Decompositions

- if relation $R(X)$ is decomposed into $R_1(X_1)$ and $R_2(X_2)$
- its lossless if $X_1 \cap X_2$ contains a key in either $R_1$ or $R_2$

Lossless means we can get back the original relation before decomposition

- if two relations don’t share a key ...
- we could get back more rows than the original

Example 1: Is this a lossless decomposition? ... YES!
- EmpDept(eid, name, dept) and Dept(dept, dept_name)

Example 2: Is this a lossless decomposition? ... YES!
- ZipCode(zip, state) and Location(addr, city, zip)

Example 3: Is it dependency preserving? ... NO!
- with original FDs: addr, city, state $\rightarrow$ zip, and zip $\rightarrow$ state

For example:

<table>
<thead>
<tr>
<th>ZipCode</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>zip</td>
<td>addr</td>
</tr>
<tr>
<td>state</td>
<td>city</td>
</tr>
<tr>
<td>99202</td>
<td>42 1st Ave</td>
</tr>
<tr>
<td>WA</td>
<td>Spokane</td>
</tr>
<tr>
<td>99203</td>
<td>42 1st Ave</td>
</tr>
<tr>
<td>WA</td>
<td>Spokane</td>
</tr>
</tbody>
</table>

“Denormalizing” gives:

<table>
<thead>
<tr>
<th>addr</th>
<th>city</th>
<th>state</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>42 1st Ave</td>
<td>Spokane</td>
<td>WA</td>
<td>99202</td>
</tr>
<tr>
<td>42 1st Ave</td>
<td>Spokane</td>
<td>WA</td>
<td>99203</td>
</tr>
</tbody>
</table>

Which violates the original (address, city, state) functional dependency
- So: BCNF but not a “good” decomposition (best we can do is 3NF)
More on Dependency Preservation

Often have to “reason” over resulting dependencies (to check preservation)

(1). Original Table
• \( R(a, b, c, d, e) \)  
  
  FDs: \( ab \rightarrow cde \) (key), \( b \rightarrow d \), and \( d \rightarrow e \)
• What NF is \( R \) in? ... 1NF because of \( b \rightarrow d \)

(2). Decomposition: \( R_1(a, b, c) \), \( R_2(b, d) \), \( R_3(d, e) \)
• Q: Is this a lossless decomposition? ... YES!

(3). To check if dependency preserving
• \( ab \rightarrow c \) from \( R_1 \), \( b \rightarrow d \) from \( R_2 \), and \( d \rightarrow e \) from \( R_3 \)
• But what about the original key? ...
• From \( ab \rightarrow c \) in \( R_1 \) and \( b \rightarrow d \) in \( R_2 \), know \( ab \rightarrow cd \)
• With \( d \rightarrow e \) in \( R_3 \), we get \( ab \rightarrow cde \) via transitivity (\( ab \rightarrow cd \) and \( d \rightarrow e \))

Summary

1. Identify Functional Dependencies (FDs) in your relations
2. See if any of them are “bad” FDs (non-trivial, non-key)
3. Decompose to get rid of “bad” FDs
   • but watch out for dependencies to a part of a candidate key (3NF)
   • also note that order of decompositions can impact results
   • “BCNF” algo. guarantees lossless but not FD preserving decompositions
4. Check that your decomposition is lossless and dependency preserving

(*) Note that we have discussed these steps “informally” ...
Conceptual Data Modeling

Design typically involves multiple steps prior to creating tables!

• We’ll focus on the **Entity-Relationship Model** (ER)
• Similar to UML diagrams in software design
• Sometimes called ER diagrams, or ERDs

### ER Basics: Entities

**Entity**
- an object distinguishable from other objects (e.g., employee “Alice”)
- described using a set of attribute-value pairs
- designated attribute (or attributes) as an “identifier” (key)
**ER Basics: Entity Sets**

**Entity Set**
- collection of similar entities (based on attributes and relationships)
- an entity is an instance (or member) of an entity set
- denoted using rectangular boxes (with attributes as ovals)

**ER Basics: Relationships**

**Relationship**
- an association among 2 or more entities
- e.g., Alice’s home department is Engineering
Relationship Set ... aka Relationship Types

- collection of similar relationships (e.g., the Manager relations)
- defined by the participating entity types and other constraints
- a relationship is an instance (or member) of a relationship set
- denoted by a diamond shape ... note: no directionality!

S. Bowers  CPSC 321, Fall 2022  11