1. 3NF vs Boyce-Codd Normal Form (Already covered)

2. Decompositions
2. DECOMPOSITIONS
1. We saw that **redundancies** in the data (“bad FDs”) can lead to data anomalies

2. We developed mechanisms to detect and remove redundancies by decomposing tables into BCNF
   1. BCNF decomposition is *standard practice*—very powerful & widely used!

3. However, sometimes decompositions can lead to **more subtle unwanted effects**…

When does this happen?
Decompositions in General

\[ R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \]

\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]

\[ R_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\[ R_1 = \text{the projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m \]

\[ R_2 = \text{the projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p \]
Theory of Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

Sometimes a decomposition is “correct”

i.e. it is a **Lossless decomposition**
## Lossy Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

What’s wrong here?

However sometimes it isn’t
A decomposition $R$ to $(R_1, R_2)$ is **lossless** if $R = R_1 \text{ Join } R_2$
Lossless Decompositions

BCNF decomposition is always lossless. Why?

If \( \{A_1, \ldots, A_n\} \rightarrow \{B_1, \ldots, B_m\} \)
Then the decomposition is lossless

Note: don’t need
\( \{A_1, \ldots, A_n\} \rightarrow \{C_1, \ldots, C_p\} \)
A relation TEACH that is in 3NF but not in BCNF

- Two FDs exist in the relation TEACH:
  - \{student, course\} \rightarrow instructor
  - instructor \rightarrow course

- \{student, course\} is a candidate key for this relation

- So this relation is in 3NF but not in BCNF

- A relation NOT in BCNF should be decomposed
Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  - D1: \{student, instructor\} and \{student, course\}
  - D2: \{course, instructor\} and \{course, student\}
  - D3: \{instructor, course\} and \{instructor, student\}
A problem with BCNF

**Problem**: To enforce a FD, must reconstruct original relation—*on each insert!*
A Problem with BCNF

We do a BCNF decomposition on a “bad” FD:
\[
\{\text{Unit}\}^+ = \{\text{Unit, Company}\}
\]

We lose the FD \[
\{\text{Company, Product}\} \rightarrow \{\text{Unit}\}!!
\]
So Why is that a Problem?

No problem so far. All *local* FD’s are satisfied.

Let’s put all the data back into a single table again:

Violates the FD \{\text{Company,Product}\} \rightarrow \{\text{Unit}\}!!
The Problem

• We started with a table R and FDs F

• We decomposed R into BCNF tables $R_1$, $R_2$, … with their own FDs $F_1$, $F_2$, …

• We insert some tuples into each of the relations—which satisfy their local FDs but when reconstruct it violates some FD across tables!

Practical Problem: To enforce FD, must reconstruct R—on each insert!
Possible Solutions

• Various ways to handle so that decompositions are all lossless / no FDs lost
  – For example 3NF- stop short of full BCNF decompositions.

• Usually a tradeoff between redundancy / data anomalies and FD preservation…

BCNF still most common- with additional steps to keep track of lost FDs...
Summary

• Constraints allow one to reason about redundancy in the data

• Normal forms describe how to remove this redundancy by decomposing relations
  – Elegant—by representing data appropriately certain errors are essentially impossible
  – For FDs, BCNF is the normal form.

• A tradeoff for insert performance: 3NF
Acknowledgement

• Some of the slides in this presentation are taken from the slides provided by the authors.
• Many of these slides are taken from cs145 course offered by Stanford University.
• Thanks to YouTube, especially to Dr. Daniel Soper for his useful videos.