Normalization

• Normalization: Improving DB by Decomposing Tables
• Bad relations have
  – anomalies
  – redundancies
  – nulls
• Solve by decomposing large relation with anomalies and redundancies into smaller ones.
• Get back original information by joining on appropriate field

Decomposition

• Decomposing relations uses projection.
• 2 new relations share one field for Join
  – If the join field is not appropriate the join will lose information.
  – Called Lossy Join because join produces invalid (SPURIOUS) tuples.
Poorly Designed Relation

<table>
<thead>
<tr>
<th>EMP</th>
<th>PROJ</th>
<th>HOURS</th>
<th>ENAME</th>
<th>PNAME</th>
<th>PLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>1</td>
<td>32.5</td>
<td>Smith</td>
<td>ProdX</td>
<td>Bell</td>
</tr>
<tr>
<td>123</td>
<td>2</td>
<td>7.5</td>
<td>Smith</td>
<td>ProdY</td>
<td>Suga</td>
</tr>
<tr>
<td>453</td>
<td>1</td>
<td>20.0</td>
<td>English</td>
<td>ProdX</td>
<td>Bell</td>
</tr>
<tr>
<td>453</td>
<td>2</td>
<td>20.0</td>
<td>English</td>
<td>ProdY</td>
<td>Suga</td>
</tr>
</tbody>
</table>

• Could add ('123', 3, ..., 'English', ...)
• To update PLOC affects multiple rows
• Cannot add project w/o employee
• Could delete Project info with last worker assigned

Bad Example of Fix-up

• Next comes an example of how not to do it.
• Relation is decomposed with wrong field in common.
• Join on that field will produce extra tuples.
Emp_Proj Broken into 2

- Join on PLOC (!) to try to recover original
- PLOC is a bad choice.

Summary

- Decomposition is not arbitrary
- Choice of ‘Pivot’ field is crucial
- Wrong choice leads to false data: Spurious Tuples
Result of Join on PLOC

Find all matches on ploc for this tuple

- 4 of these tuples are bogus (Spurious)
- PLOC was a bad choice!

Normalization Issues

- When is a table defective?
- How to choose the right pivot?
- To answer these questions
  - Detour through Functional Dependency Theory.
  - Use theory to detect and fix defective tables.