CMSC424: Normalization

Instructor: Amol Deshpande
amol@cs.umd.edu
Where did we come up with the schema that we used?
  ◦ E.g. why not store the actor names with movies?

If from an E-R diagram, then:
  ◦ Did we make the right decisions with the E-R diagram?

Goals:
  ◦ Formal definition of what it means to be a “good” schema.
  ◦ How to achieve it.
Movie(title, year, length, inColor, studioName, producerC#, starName)

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>inColor</th>
<th>StudioName</th>
<th>prodC#</th>
<th>StarName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>Hamill</td>
</tr>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>Fisher</td>
</tr>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>H. Ford</td>
</tr>
<tr>
<td>King Kong</td>
<td>2005</td>
<td>187</td>
<td>Yes</td>
<td>Universal</td>
<td>150</td>
<td>Watts</td>
</tr>
<tr>
<td>King Kong</td>
<td>1933</td>
<td>100</td>
<td>no</td>
<td>RKO</td>
<td>20</td>
<td>Fay</td>
</tr>
</tbody>
</table>

**Issues:**

1. Redundancy ➔ higher storage, inconsistencies (“anomalies”)
   *update anomalies, insertion anomalies*

2. Need nulls
   Unable to represent some information without using nulls
   *How to store movies w/o actors (pre-productions etc) ?*

**Key Problem:**

There are functional dependencies: A ➔ B such that A is duplicated

*Definition: A ➔ B: If two tuples agree on A, they agree on B*

Therefore: If A is duplicated, B is duplicated
Movie($title$, $year$, length, inColor, studioName, producerC#, starNames)

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>inColor</th>
<th>StudioName</th>
<th>prodC#</th>
<th>StarNames</th>
</tr>
</thead>
<tbody>
<tr>
<td>Star wars</td>
<td>1977</td>
<td>121</td>
<td>Yes</td>
<td>Fox</td>
<td>128</td>
<td>{Hamill, Fisher, H. ford}</td>
</tr>
<tr>
<td>King Kong</td>
<td>2005</td>
<td>187</td>
<td>Yes</td>
<td>Universal</td>
<td>150</td>
<td>Watts</td>
</tr>
<tr>
<td>King Kong</td>
<td>1933</td>
<td>100</td>
<td>no</td>
<td>RKO</td>
<td>20</td>
<td>Fay</td>
</tr>
</tbody>
</table>

**Issues:**

3. Avoid sets
   - Hard to represent
   - Hard to query
Smaller schemas always good ????

Split Studio \((name, address, \text{presC#})\) into:

- Studio1 \((name, \text{presC#})\)
- Studio2 \((name, address)\)

<table>
<thead>
<tr>
<th>Name</th>
<th>\text{presC#}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fox</td>
<td>101</td>
</tr>
<tr>
<td>Studio2</td>
<td>101</td>
</tr>
<tr>
<td>Universal</td>
<td>102</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fox</td>
<td>Address1</td>
</tr>
<tr>
<td>Studio2</td>
<td>Address1</td>
</tr>
<tr>
<td>Universal</td>
<td>Address2</td>
</tr>
</tbody>
</table>

This process is also called “\textit{decomposition}”

\textbf{Issues:}

4. Requires more joins (w/o any obvious benefits)

5. Hard to check for some dependencies

What if the “address” is actually the \text{presC#’s} address ?

No easy way to ensure that constraint (w/o a join).
**Issues:**

6. “joining” them back results in more tuples than what we started with  
   (King Kong, 1933, Watts) & (King Kong, 2005, Faye)

   This is a “lossy” decomposition

   We lost some constraints/information

   The previous example was a “lossless” decomposition.
Desired data

- No sets
- Correct and faithful to the original design
  - Avoid lossy decompositions
- As little redundancy as possible
  - To avoid potential anomalies
- No “inability to represent information”
  - Nulls shouldn’t be required to store information
- Dependency preservation
  - Should be possible to check for constraints

Not always possible. We sometimes relax these for:

* simpler schemas, and fewer joins during queries.*
Approach

1. We will encode and list all our knowledge about the schema
   ◦ Functional dependencies (FDs)
     \[ \text{SSN} \rightarrow \text{name} \] (means: SSN “implies” length)
   ◦ If two tuples have the same “SSN”, they must have the same “name”
     \[ \text{movietitle} \rightarrow \text{length} \] Not true.
   ◦ But, \((\text{movietitle}, \text{movieYear}) \rightarrow \text{length}\) --- True.

2. We will define a set of rules that the schema must follow to be considered good
   ◦ “Normal forms”: 1NF, 2NF, 3NF, BCNF, 4NF, ...
   ◦ A normal form specifies constraints on the schemas and FDs

3. If not in a “normal form”, we modify the schema
Rest...

- See 424 Normalization Slides or a textbook