Introduction to Databases, Fall 2003
IT University of Copenhagen

Lecture 6, part I: Normalization II

September 30, 2003

Lecturer: Rasmus Pagh
Today’s lecture

- What you should remember from previously.
- Multivalued dependencies.
- 4th normal form.
- Some observations on normalization.
In this lecture I will assume that you remember:

- Concepts of normalization:
  - Decomposition
  - Functional dependency
  - Boyce-Codd normal form and 3rd normal form
Next: Multivalued dependencies.
Redundancy in BCNF relations

Boyce-Codd normal form eliminates redundancy in each tuple, but may leave redundancy among tuples in a relation.

This happens, for example, if two many-many relationships are represented in a relation.

[Figure 3.29 shown on slide]

**Example:** In the relation $\text{StarsIn}(\text{name}, \text{street}, \text{city}, \text{title}, \text{year})$ we could represent two many-many relationships: between actors and addresses, and between actors and movies.
Then what about something like one of these:

<table>
<thead>
<tr>
<th>name</th>
<th>street</th>
<th>city</th>
<th>title</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Fisher</td>
<td>123 Maple St.</td>
<td>Hollywood</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>5 Locust Ln.</td>
<td>Malibu</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>NULL</td>
<td>NULL</td>
<td>Star Wars</td>
<td>1977</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>NULL</td>
<td>NULL</td>
<td>Empire Strikes Back</td>
<td>1980</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>NULL</td>
<td>NULL</td>
<td>Return of the Jedi</td>
<td>1983</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>street</th>
<th>city</th>
<th>title</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Fisher</td>
<td>123 Maple St.</td>
<td>Hollywood</td>
<td>Star Wars</td>
<td>1977</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>5 Locust Ln.</td>
<td>Malibu</td>
<td>Empire Strikes Back</td>
<td>1980</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>NULL</td>
<td>NULL</td>
<td>Return of the Jedi</td>
<td>1983</td>
</tr>
</tbody>
</table>
A better idea is to eliminate redundancy by decomposing StarsIn as follows:

<table>
<thead>
<tr>
<th>name</th>
<th>street</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Fisher</td>
<td>123 Maple St.</td>
<td>Hollywood</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>5 Locust Ln.</td>
<td>Malibu</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>name</th>
<th>title</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>C. Fisher</td>
<td>Star Wars</td>
<td>1977</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>Empire Strikes Back</td>
<td>1980</td>
</tr>
<tr>
<td>C. Fisher</td>
<td>Return of the Jedi</td>
<td>1983</td>
</tr>
</tbody>
</table>
When can we decompose?

When can we decompose a relation R? Suppose we decompose into two relations (for simplicity we assume that there is just one common attribute):

\[ R_1(A, B_1, B_2, \ldots, B_m) \]
\[ R_2(A, C_1, C_2, \ldots, C_k) \]

Now consider a specific value \( a \) for attribute A, occurring in the set of tuples \( T_1 \) from \( R_1 \) and in the set of tuples \( T_2 \) from \( R_2 \).

When we join \( R_1 \) and \( R_2 \), every pair of tuples from \( T_1 \) and \( T_2 \) are combined.
When can we decompose (2)?

Example:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>U</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>E</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>NW</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SE</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>SW</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>45</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>90</td>
<td></td>
</tr>
</tbody>
</table>
When we can decompose $R$ into relations

$$R_1(A_1, A_2, \ldots A_n, B_1, B_2, \ldots, B_m)$$

$$R_2(A_1, A_2, \ldots A_n, C_1, C_2, \ldots, C_k)$$

(with no Bs among the Cs) then we say that there is a

**multivalued dependency (MVD)** from the As to the Bs, written

$$A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$$

**Example:** Since $\text{StarsIn}$ can be decomposed into

$\text{StarsIn}_1(name, street, city)$ and $\text{StarsIn}_2(name, title, year)$

it has the MVD $name \rightarrow street \ \text{city}$. 
Multi-valued dependencies, book’s definition

A1 A2...An → B1 B2...Bm

holds exactly if:

For every pair of tuples \( t \) and \( u \) from R that agree on all As, we can find some tuple \( v \) in R that agrees:

- With both \( t \) and \( u \) on the As
- With \( t \) on the Bs
- With \( u \) on the Cs

[Figure 3.30 shown on slide]

Problem session (5 minutes): Convince yourselves that this definition, used in the coursebook, is equivalent to the one given previously in this lecture.
If \( \{A_1, A_2, \ldots, A_n\} \) form a superkey, then for any \( B_1, B_2, \ldots, B_m \) we unavoidably have:

\[
A_1\ A_2\ldots A_n \rightarrow B_1\ B_2\ldots B_m
\]

An MVD is said to be **trivial** if either

- One of the Bs is among the As, or
- All the attributes of R are among the As and Bs.
Next: 4th normal form.
4th normal form

Roughly speaking, a relation is in 4th normal form if it cannot be meaningfully decomposed into two relations. More precisely:

A relation is in **fourth normal form** (4NF) if any multivalued dependency among its attributes is either unavoidable or trivial.

**Example:** StarsIn has the MVD name $\rightarrow$ street city which is nontrivial. Since name is not a superkey the relation is not in 4NF.
Decomposing a relation into 4NF

Suppose we have a relation $R$ which is not in 4NF. Then there is a nontrivial MVD

$$A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$$

which is not unavoidable.

To eliminate the MVD we split $R$ into two relations:

- One with all attributes of $R$ except $B_1, B_2, \ldots, B_m$.
- One with attributes $A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m$.

If any of the resulting relations is not in 4NF, the process is repeated.
Recall the relation StarsIn with schema
StarsIn(name, street, city, title, year)

It has the following nontrivial MVD, which is not unavoidable:

\[ \text{name } \rightarrow\text{ street city} \]

Thus the decomposition yields the following relations (both in 4NF):
StarsIn1(name, street, city)
StarsIn2(name, title, year)
Problem session (5 minutes)

What would happen if we tried to do the decomposition:

- According to an unavoidable MVD?
- According to an MVD including all attributes of R?
- According to an MVD with a common attribute on the left and right hand side?
Reasoning about MVDs

As for functional dependencies, there are rules that can be used to derive new MVDs from a set of already known MVDs:

- **The trivial dependencies rule.** If \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \) then \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m C_1 \ldots C_k \), where the \( C \)'s are among the \( A \)'s.

- **The transitive rule.** If \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \) and \( B_1 B_2 \ldots B_m \rightarrow C_1, \ldots, C_k \), where none of the \( C \)'s are among the \( B \)'s, then

\[
A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k
\]

- **The complementation rule.** If \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \) then \( A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k \), where the \( C \)'s are those attributes not among the \( A \)'s or \( B \)'s.
Next: Some observations on normalization
Relationship among normal forms

Inclusion among normal forms:

Any relation in 4NF is also in BCNF.
Any relation in BCNF is also in 3NF.

[Figure 3.31 shown on slide]

Properties of normal forms:

A “higher” normal form has less redundancy, but may not preserve functional and multivalued dependencies.

[Figure 3.32 shown on slide]
How should normal forms be used?

The various normal forms may be seen as *guidelines* for designing a good relation schema. Some complexities that arise are:

- Should we split keys, introducing dependencies between relations (in 3NF we do not)?
- What is the effect of decomposition on performance?
- How does decomposition affect query programming?
As a minimum, you should after this week:

- Be able to determine whether a relation is in 4th normal form.
- Be able to split a relation in several relations to achieve 4th normal form.