Algorithms for decomposition

Introduction to Database Design 2011, Lecture 9
Overview

• Decomposition to BCNF
  - algorithm for lossless decomposition

• Decomposition to 3NF
  - algorithm for lossless and dependency preserving decomposition

• 4NF

• Course evaluation
Mandatory assignments

• Final deadline for mandatory assignments 1-3 on April 12
BCNF decomposition

- Compute $F^+$
- Repeat the following while the schema is not BCNF
  - Find a BCNF violation $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ in schema $R(\alpha)$
  - Decompose $R$ into $((\alpha - B_1 B_2 \ldots B_m) \cup A_1 A_2 \ldots A_n)$ and $(A_1 A_2 \ldots A_n B_1 B_2 \ldots B_m)$
Example

• Decompose the relation

\[ cd\_shop(cd\_id, \text{artist}, \text{title}, \text{order\_id}, \text{order\_date}, \text{quantity}, \text{customer\_id}, \text{name}, \text{address}) \]

• With the functional dependencies

\[
\begin{align*}
\text{cd\_id} & \rightarrow \text{artist, title} \\
\text{customer\_id} & \rightarrow \text{name, address} \\
\text{order\_id} & \rightarrow \text{order\_date, customer\_id} \\
\text{order\_id, cd\_id} & \rightarrow \text{quantity}
\end{align*}
\]
Non determinancy

• Much depends on the choice of BCNF violation

• Try e.g. decomposing first using

  \[ order_id \rightarrow order_date, customer_id \]

• There is no guarantee that decomposition is dependency preserving

• (even if there is a dependency preserving decomposition)

• One heuristic is to maximise right hand sides of BCNF violations
Correctness

• Tables become smaller for every decomposition
• Every 2-attribute table is BCNF
• So in the end, the schema must be BCNF
• Every decomposition is lossless by rule mentioned 2 weeks ago (book page 346)
3NF decomposition

• Compute a canonical cover

• Create a table \((A_1 A_2 \ldots A_n B_1 B_2 \ldots B_n)\) for every dependency \(A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_n\) in cover

• If no table contains a candidate key
  - add a table whose attributes is a candidate key

• Optional: erase unnecessary tables
Example

- Decompose the relation

\[
\text{cd\_shop}(cd\_id, \text{artist}, \text{title, order\_id, order\_date, quantity, customer\_id, name, address})
\]

- With the functional dependencies

\[
\begin{align*}
\text{cd\_id} & \rightarrow \text{artist, title} \\
\text{customer\_id} & \rightarrow \text{name, address} \\
\text{order\_id} & \rightarrow \text{order\_date, customer\_id} \\
\text{order\_id}, \text{cd\_id} & \rightarrow \text{quantity}
\end{align*}
\]

- (note that these are a canonical cover)
Alternative BCNF decomposition

• Example suggests the following alternative algorithm for BCNF decomposition
  - Use 3NF decomposition
  - Do further BCNF decompositions if needed
3NF decomposition examples

- `dept_advisor(s_ID, i_ID, dept_name)`

  
  \[
  \begin{array}{c}
  i_ID \rightarrow dept\_name \\
  s_ID, dept\_name \rightarrow i_ID \\
  \end{array}
  \]

- Variant: `dept_advisor(s_ID, i_ID, dept_name, semester)` (same dependencies)
Example

- Employee of the month example for Big Kahuna Burger
- Table: \((\text{empl}_\text{id}, \text{name}, \text{branch}, \text{year}, \text{month})\)
- Functional dependencies:
  \[
  \text{empl}_\text{id} \rightarrow \text{name}, \text{branch} \\
  \text{branch}, \text{year}, \text{month} \rightarrow \text{empl}_\text{id}
  \]
- Decompose to BCNF and to 3NF
Correctness

• Decomposition is lossless:
  - At least one schema contains candidate key
  - Losslessness follows from generalisation of “losslessness rule”

• Decomposition is dependency preserving
  - Each dependency in cover can be checked on one relation

• For proof of 3NF see book (slightly difficult)
On using the decomposition algorithms

- Could use decomposition to design databases
- First find all necessary attributes and functional dependencies
- Decompose to 3NF or BCNF
- I do not recommend this!
- Much better to think in terms of entities and relations
- But algorithms are good to know if you encounter redundancy problems
4NF
Example

- Consider a database storing information about movie stars

<table>
<thead>
<tr>
<th>name</th>
<th>address</th>
<th>movie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samuel L Jackson</td>
<td>Sunshine Blvd 1</td>
<td>Pulp Fiction</td>
</tr>
<tr>
<td>Samuel L Jackson</td>
<td>Rainy Street 134</td>
<td>Pulp Fiction</td>
</tr>
<tr>
<td>Samuel L Jackson</td>
<td>Sunshine Blvd 1</td>
<td>Snakes on a Plane</td>
</tr>
<tr>
<td>Samuel L Jackson</td>
<td>Rainy Street 134</td>
<td>Snakes on a Plane</td>
</tr>
</tbody>
</table>

- Clearly lots of redundancy here
- But no non-trivial functional dependencies!
- So BCNF
Problem in a nutshell

- Attributes *address* and *movie* are independent and not determined by other attributes

- For every pair of tuples

  +----------------+----------------+-------+
  | name           | address        | movie |
  +----------------+----------------+-------+
  | Actor name     | Address 1      | Movie 1|
  | Actor name     | Address 2      | Movie 2|
  +----------------+----------------+-------+

- There are also tuples

  +----------------+----------------+-------+
  | name           | address        | movie |
  +----------------+----------------+-------+
  | Actor name     | Address 1      | Movie 2|
  | Actor name     | Address 2      | Movie 1|
  +----------------+----------------+-------+

- This is called a **multivalued dependency**
Multivalued dependencies

- Consider a table $R(\alpha \beta \gamma)$

**Definition.** There is a multivalued dependency $\alpha \Rightarrow \beta$ if for all tuples $t, u$ in all legal instances

- if $t[\alpha] = u[\alpha]$

- then there exists tuple $s$ such that

  $s[\alpha] = t[\alpha]$

  $s[\beta] = t[\beta]$

  $s[\gamma] = u[\gamma]$

- In example $name \Rightarrow address$
Rules for multivalued dependencies

- In $R(\alpha\beta\gamma)$ if $\alpha \rightarrow^* \beta$ then also $\alpha \rightarrow^* \gamma$
- If $\alpha \rightarrow^* \beta$ then also $\alpha \rightarrow^* \beta$ (can take $s = u$)
- Consequences
  - if $\beta \subseteq \alpha$ then $\alpha \rightarrow^* \beta$
  - if $\alpha$ superkey then $\alpha \rightarrow^* \beta$
- **Transitivity**: if $\alpha \rightarrow^* \beta$ and $\beta \rightarrow^* \gamma$ then $\alpha \rightarrow^* \gamma$
- It is **not** the case that if $\alpha \rightarrow^* \beta\gamma$ then $\alpha \rightarrow^* \beta$
• **Definition.** A table $r(R)$ is in **4NF** if for all multivalued dependencies $\alpha \rightarrow \beta$ either
  - $\beta \subseteq \alpha$ ($\alpha \rightarrow \beta$ is trivial)
  - or $\alpha$ is a superkey

• **Definition.** A schema is in **4NF** if all tables are in **4NF**

• **Theorem.** A schema in **4NF** is also **BCNF**
4NF decomposition

- There is a lossless decomposition algorithm for 4NF
- It is the same as the one for BCNF but uses multivalued dependencies
Normal forms

• A tower of normal forms
  - 4NF
  - BCNF
  - 3NF
  - 2NF
  - 1NF

• Any schema satisfying a normal form also satisfies the ones below

• (there do exist even higher normal forms)
Summary

• Algorithm for lossless decomposition into BCNF

• Algorithm for lossless and dependency preserving decomposition into 3NF

• Even BCNF schemes may have redundancy

• 4NF normalisation gets rid of even more redundancy