Normalization Algorithms

CS157A
Chris Pollett
Nov. 21, 2005.
Outline

- General Definitions of Normal Forms
- Boyce-Codd Normal Form
- Properties of Relational Decomposition
General Definitions of Normal Forms

• The original definitions of normal forms we gave last day were defined in terms of the primary key.

• Today, we’ll briefly consider definitions which are with respect to any key.
General Definition 2NF

• A relation schema R is in 2NF if every nonprime attribute A in R is not partially dependent on any key of R.
• For example, suppose we had the table:
  LOTS(PropertyID, CountyName, Number, Area, Price, TaxRate)
  and we had the functional dependencies:
  PropertyID --> CountyName, Number, Area, Price, TaxRate
  CountyName, Number --> PropertyID, Area, Price, TaxRate
  CountyName --> TaxRate
  Area --> Price
• In this situation, TaxRate is nonprime and partially depends on the candidate key CountryName, Number.
• To put this in 2NF, we would set up a new relation for each partial key with its dependent attributes.
General Definition 3NF

- Recall a *trivial dependency* is a dependency of the form $X \rightarrow A$ where $X \supseteq A$.

- A relation schema $R$ is in 3NF if whenever a nontrivial FD $X \rightarrow A$ holds in $R$, either (a) $X$ is a superkey of $R$ or (b) $A$ is a prime attribute of $R$.

- For instance, if we had split LOTS from the last slide into:
  
  LOTS1(\textbf{PropertyID}, \text{CountyName}, \text{Number}, \text{Area}, \text{Price})
  
  LOTS2(\text{CountyName}, \text{TaxRate})
  
  It would be in 2NF but not 3NF, because Area $\rightarrow$ Price in LOTS1 but Area is not a superkey and Price is not prime.

- Conditions (a) and (b) guarantee both being in 2NF and there being no transitive dependencies.
Boyce-Codd Normal Form

• The general definition of 3NF can be further simplified to give an even stronger normal form:

• A relation schema \textit{R is in Boyce-Codd Normal Formal (BCNF)} if whenever a nontrivial FD \(X \rightarrow A\) holds in \(R\), then \(X\) is a superkey of \(R\).

• For example, suppose we put \textit{LOTS1} of the last slide into 3NF by making two relation schemas:
  - \textit{LOTS1a(PropertyID, CountyName, Number, Area)}
  - \textit{LOTS1b(Area, Price)}

• Now suppose we added the FD \(Area \rightarrow CountyName\), since maybe from the plots size we can figure out which county we are in.

• Even with this new FD our decomposition is still in 3NF.

• However, it is not in BCNF.
Properties of Relational Decomposition

• We are now about to describe algorithms for putting our tables in normal forms.
• We assume we start with a *universal relation* which has all the attributes of the DB we want to store.
• We assume we also have a list of FDs, F, for these attributes.
• The goal is to decompose this universal relation \( R = (A_1, \ldots, A_n) \) into relation schemas \( D = \{R_1, \ldots, R_m\} \) in some normal form.
• We also want the decomposition to be *attribute preserving*:
  \[
  \bigcup_{i=1}^{m} R_i = R
  \]
• We also describe some other properties for the decomposition on the next couple of slides.
Dependency Preservation

- FDs contain important semantics of the data.
- It is useful that for each FD $X \rightarrow Y$ our decomposition either has a table $R_i$ which contains the attributes of $X$ and $Y$ or this dependency is inferable from the other dependencies that do appear.
- Let $\pi_{R_i}(F)$ denote those FDs in $F^+$ all of whose attributes lie in $R_i$.
- $D = \{R_1, \ldots, R_m\}$ is dependency preserving if:
  
  $\left((\pi_{R_1}(F)) \cup \ldots \cup (\pi_{R_n}(F))\right)^+ = F^+$
Lossless Join Property

• Another property we like our decompositions to have is that we join all the tables together of our decomposition we do not get any spurious tuples.

• Let $r$ be a relation state for our universal relation $R$. Formally, $D = \{R_1, \ldots, R_m\}$ has the lossless join property if it satisfies:

$$\pi_{R_1}(r) \ast \cdots \ast \pi_{R_m}(r)$$
Testing for the Lossless Join Property

**Input:** A universal relation $R$, a decomposition $D = \{R_1, \ldots, R_m\}$ of $R$ and a set $F$ of FDs.

1. Create an initial matrix $S$ with one row $i$ for each relation $R_i$ in $D$, and one column $j$ for each attribute $A_j$ in $R$.
2. Set $S(i,j) := b_{ij}$ for all matrix entries.
3. For each row $i$ representing relation $R_i$
   {for each column $j$ with attribute $A_j$
     {if (relation $R_i$ includes attribute $A_j$) then set $S(i,j) := a_j$;};};
4. Repeat the following loop until a complete loop execution results in no change to $S$:
   {for each FD $X --> Y$ in $F$
     {for all rows in $S$ that have the same symbols in the columns corresponding to attributes in $X$
       {make the symbols in the columns that correspond to an attribute in $Y$ be the same in all these rows as follows: If any of the rows has an “a” symbol for the column, set the other rows to that same “a” symbol. If no “a” symbol exists, choose one of the “b” symbols and set the other rows to the same “b” symbol in that column };};};;
5. If a row is made up entirely of “a” symbols then the decomposition has the lossless join property.
Example

- Suppose
  \[ R = \{ \text{SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} \} \]
  \[ D = \{ R_1, R_2, R_3 \} \]
  \[ R_1 = \text{EMP} = \{ \text{SSN, ENAME} \} \]
  \[ R_2 = \text{PROJ} = \{ \text{PNUMBER, PNAME, PLOCATION} \} \]
  \[ R_3 = \text{WORKS\_ON} = \{ \text{SSN, PNUMBER, HOURS} \} \]
  \[ F = \{ \text{SSN} \rightarrow \{ \text{ENAME} \}; \text{PNUMBER} \rightarrow \{ \text{PNAME, PLOCATION} \}; \{ \text{SSN, PNUMBER} \} \rightarrow \text{HOURS} \} \]

Initially, after step 3 get:

<table>
<thead>
<tr>
<th>SSN</th>
<th>ENAME</th>
<th>PNUMBER</th>
<th>PNAME</th>
<th>PLOCATION</th>
<th>HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>a1</td>
<td>a2</td>
<td>b13</td>
<td>b14</td>
<td>b15</td>
</tr>
<tr>
<td>R2</td>
<td>b21</td>
<td>b22</td>
<td>a3</td>
<td>a4</td>
<td>a5</td>
</tr>
<tr>
<td>R3</td>
<td>a1</td>
<td>b32</td>
<td>a3</td>
<td>b34</td>
<td>b35</td>
</tr>
</tbody>
</table>

after step 4 get:

<table>
<thead>
<tr>
<th>SSN</th>
<th>ENAME</th>
<th>PNUMBER</th>
<th>PNAME</th>
<th>PLOCATION</th>
<th>HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>R1</td>
<td>a1</td>
<td>a2</td>
<td>b13</td>
<td>b14</td>
<td>b15</td>
</tr>
<tr>
<td>R2</td>
<td>b21</td>
<td>b22</td>
<td>a3</td>
<td>a4</td>
<td>a5</td>
</tr>
<tr>
<td>R3</td>
<td>a1</td>
<td>a2</td>
<td>a3</td>
<td>a4</td>
<td>a5</td>
</tr>
</tbody>
</table>
Binary Test for LJP

• A decomposition $D = \{R_1, R_2\}$ of $R$ has the lossless join property with respect to $F$ iff
  - The FD $(R_1 \cap R_2) \rightarrow (R_1 - R_2)$ is in $F^+$, or
  - The FD $(R_1 \cap R_2) \rightarrow (R_2 - R_1)$ is in $F^+$

• IF $D$ has the lossless join property and $D'$ is obtained from $D$ by replacing an $R_i$ with a decomposition $D''$ which has the LJP then $D'$ has the lossless join property.