BCNF and Normalization

Zaki Malik
October 21, 2008
Relational Schema Design

• Goal of relational schema design is to avoid redundancy and anomalies.

  - *Redundancy*: information is repeated unnecessarily in several tuples.
  - *Update anomalies*: We change information in one tuple but leave the old information in another tuple.
  - *Insertion anomalies*: It is not possible to store some information unless some other, unrelated information is stored as well.
  - *Deletion anomalies*: If a set of values becomes empty, we may lose other information as a side effect.
Bad Design

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>Export</td>
<td>Molson</td>
<td>G.I. Lager</td>
</tr>
<tr>
<td>Janeway</td>
<td>Voyager</td>
<td>G.I. Lager</td>
<td>Gr. Is.</td>
<td>G.I. Lager</td>
</tr>
<tr>
<td>Spock</td>
<td>Enterprise</td>
<td>Export</td>
<td>Molson</td>
<td>G.I. Lager</td>
</tr>
</tbody>
</table>

- **Redundancy**

- **Update anomaly**
  - if Janeway is transferred to *Intrepid*, will we remember to change each of her tuples?

- **Deletion anomaly**
  - If nobody likes Export, we lose track of the fact that Molson manufactures Export.
Another Example

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
<th>StudentName</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>E-Business</td>
<td>211 McBryde</td>
<td>32</td>
<td>Adam</td>
<td>71 Main Street</td>
</tr>
<tr>
<td>6722</td>
<td>CS</td>
<td>Advanced DB</td>
<td>210 McBryde</td>
<td>15</td>
<td>Adam</td>
<td>71 Main Street</td>
</tr>
<tr>
<td>4322</td>
<td>Electrical</td>
<td>DB</td>
<td>220 McBryde</td>
<td>29</td>
<td>Suri</td>
<td>54 Elm Street</td>
</tr>
<tr>
<td>5722</td>
<td>CS</td>
<td>DB</td>
<td>311 Durham</td>
<td>34</td>
<td>Suri</td>
<td>54 Elm Street</td>
</tr>
<tr>
<td>5722</td>
<td>CS</td>
<td>DB</td>
<td>311 Durham</td>
<td>34</td>
<td>Joe</td>
<td>33 Astoria Ave</td>
</tr>
<tr>
<td>6722</td>
<td>CS</td>
<td>Advanced DB</td>
<td>210 McBryde</td>
<td>15</td>
<td>Joe</td>
<td>33 Astoria Ave</td>
</tr>
</tbody>
</table>
Relational Decomposition

- Accepted way to eliminate anomalies is to “decompose” relations.
- Given a relation $R(A_1, A_2, \ldots, A_n)$, two relations $S(B_1, B_2, \ldots, B_m)$ and $T(C_1, C_2, \ldots, C_k)$ form a decomposition of $R$ if
  1. the attributes of $S$ and $T$ together make up the attributes of $R$, i.e.,
     \[
     \{A_1, A_2, \ldots, A_n\} = \{B_1, B_2, \ldots, B_m\} \cup \{C_1, C_2, \ldots, C_k\}.
     \]
  2. the tuples in $S$ are the projections into $\{R_1, R_2, \ldots, R_m\}$ of the tuples in $R$.
  3. the tuples in $T$ are the projections into $\{C_1, C_2, \ldots, C_k\}$ of the tuples in $R$.
  4. $S$ and $T$ do not contain duplicate tuples.
Example of Decomposition

- Decompose
  - Courses into
    - Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment) and
    - Courses2(Number, DepartmentName, StudentName, Address).
- Are the anomalies removed?
  - Redundancy
  - Update
  - Insertion
  - Deletion
Triviality of FDs

An FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is

- **trivial** if the $B$’s are a subset of the $A$’s,
  \[
  \{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}
  \]

- **non-trivial** if at least one $B$ is not among the $A$’s,
  \[
  \{B_1, B_2, \ldots B_n\} \setminus \{A_1, A_2, \ldots A_n\} \neq \emptyset
  \]

- **completely non-trivial** if none of the $B$’s are among the $A$’s, i.e.,
  \[
  \{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset.
  \]

- **Trivial dependency rule**: The FD $A_1A_2\ldots A_n \rightarrow B_1B_2\ldots B_m$ is equivalent to the FD $A_1A_2\ldots A_n \rightarrow C_1C_2\ldots C_k$, where the $C$’s are those $B$’s that are not $A$’s, i.e.,
  \[
  \{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} \setminus \{A_1, A_2, \ldots, A_n\}.
  \]

- What good are trivial and non-trivial dependencies?
  - Trivial dependencies are always true.
  - They help simplify reasoning about FDs.
Boyce-Codd Normal Form

- Condition on the FDs in a relation that guarantees that anomalies do not exist.

- A relation $R$ is in Boyce-Codd Normal Form (BCNF) if and only if for every non-trivial FD $A_1A_2\ldots A_n \rightarrow B$ for $R$, $\{A_1, A_2,\ldots, A_n\}$ is a superkey for $R$.

- Informally, the left side of every non-trivial FD must be a superkey.

- A relation $R$ violates BCNF if it has an FD such that the attributes of the left side of an FD do not form a superkey.
Closures of FDs vs. Closures of Attributes

- Both algorithms take as input a relation $R$ and a set of FDs $F$.
- **Closure of FDs:**
  - Computes $\{F\}^+$, the set of all FDs that follow from $F$.
  - Output is a set of FDs.
  - Output may contain an exponential number of FDs.
- **Closure of attributes:**
  - In addition, takes a set $\{A_1, A_2, \ldots, A_n\}$ of attributes as input.
  - Computes $\{A_1, A_2, \ldots, A_n\}^+$, the set of all attributes $B$ such that the $A_1A_2\ldots A_n \rightarrow B$ follows from $F$.
  - Output is a set of attributes.
  - Output may contain at most the number of attributes in $R$. 
Checking for BCNF Violations

- List all FDs.
- Ensure that left hand side of each FD is a superkey.
- We have to first find all the keys!
- Is Courses(Number, DepartmentName, CourseName, Classroom, Enrollment, StudentName, Address) in BCNF?
- FDs are
  
  \[
  \begin{align*}
  \text{Number DepartmentName} & \rightarrow \text{CourseName} \\
  \text{Number DepartmentName} & \rightarrow \text{Classroom} \\
  \text{Number DepartmentName} & \rightarrow \text{Enrollment}
  \end{align*}
  \]
- What is \(\{\text{Number, DepartmentName}\}^+\)?
  \[
  \{\text{Number, DepartmentName, CourseName, Classroom, Enrollment}\}
  \]
- Therefore, the key is
  \[
  \{\text{Number, DepartmentName, StudentName, Address}\}
  \]
- The relation is not in BCNF.
Decomposition into BCNF

- Suppose $R$ is a relation schema that violates BCNF.
- We can decompose $R$ into a set $S$ of new relations such that
  1. each relation in $S$ is in BCNF and
  2. we can “recover” $R$ from the relations in $S$, i.e., the relations in $S$ “faithfully” represent the data in $R$.
- Let $X$ be the set of all attributes of $R$.
- Suppose the FD $A_1A_2\ldots A_m \rightarrow B$ violates BCNF.
- Decomposition algorithm:
  1. Compute $\{A_1A_2\ldots A_m\}^+$ and augment the FD to $A_1A_2\ldots A_m \rightarrow \{A_1, A_2\ldots, A_m\}^+$.
  2. Decompose $R$ into two relations containing
     2.1 all the attributes in $\{A_1, A_2\ldots, A_m\}^+$
     2.2 all the attributes on the left side of the FD and all the attributes of $R$ not on the right side of the FD, i.e., $X - \{A_1, A_2\ldots, A_m\}^+ \cup \{A_1, A_2\ldots, A_m\}$.
  3. Find FDs in the new relations and decompose them if they are not in BCNF.
Decomposing Courses

- Schema is Courses(Number, DepartmentName, CourseName, Classroom, Enrollment, StudentName, Address).
- BCNF-violating FD is
  \[ \text{Number DepartmentName} \rightarrow \text{CourseName Classroom Enrollment} \]
  - What is \( \{\text{Number, DepartmentName}\}^+ \)?
  \[ \{\text{Number, DepartmentName, CourseName, Classroom, Enrollment}\} \]

- Decompose Courses into
  Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment) and
  Courses2(Number, DepartmentName, StudentName, Address).
Decomposing Courses

- Decompose Courses into
  Courses1(Number, DepartmentName, CourseName, Classroom, Enrollment) and
  Courses2(Number, DepartmentName, StudentName, Address).

<table>
<thead>
<tr>
<th>Number</th>
<th>DeptName</th>
<th>CourseName</th>
<th>Classroom</th>
<th>Enrollment</th>
</tr>
</thead>
<tbody>
<tr>
<td>4604</td>
<td>CS</td>
<td>E-Business</td>
<td>211 McBryde</td>
<td>32</td>
</tr>
<tr>
<td>6722</td>
<td>CS</td>
<td>Advanced DB</td>
<td>210 McBryde</td>
<td>15</td>
</tr>
<tr>
<td>4322</td>
<td>Electrical</td>
<td>DB</td>
<td>220 McBryde</td>
<td>29</td>
</tr>
<tr>
<td>5722</td>
<td>CS</td>
<td>DB</td>
<td>311 Durham</td>
<td>34</td>
</tr>
</tbody>
</table>

- Are there any BCNF violations in the two new relations?
Another Example of Decomposition

- Schema is Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId)
- What are the FDs?
  - ID $\rightarrow$ Name FavouriteAdvisorId
  - AdvisorId $\rightarrow$ AdvisorName
- What is the key? \{ID, AdvisorId\}
- Is there a BCNF violation? Yes.
- Use ID $\rightarrow$ Name Level FavouriteAdvisorId to decompose.
  - $\{ID\}^+$ is \{ID, Name, FavouriteAdvisorId\}
  - Schemas for new relations are
    - Students1(ID, Name, FavouriteAdvisorId)
    - Students2(ID, AdvisorId, AdvisorName)
Another Example of Decomposition (2)

- What are the FDs in \textbf{Student1}(ID, Name, FavouriteAdvisorId)?
  There are none that violate BCNF.

- What are the FDs in \textbf{Students2}(ID, AdvisorId, AdvisorName)?
  - AdvisorId $\rightarrow$ AdvisorName

- Repeat the decomposition process.

- Use AdvisorId $\rightarrow$ AdvisorName to decompose.
  - $\{\text{AdvisorId}\}^+$ is $\{\text{AdvisorId, AdvisorName}\}$
  - Schemas for new relations are
    - \textbf{Students2}(ID, AdvisorId)
    - \textbf{Students3}(AdvisorId, AdvisorName)
BCNFs and Two-Attribute Relationships

- True or False: Every two-attribute relation $R(A, B)$ is in BCNF.

- The statement is true. Why?

- Consider four possible cases:
  1. There are no non-trivial FDs.
  2. $A \rightarrow B$ is the only non-trivial FD.
  3. $B \rightarrow A$ is the only non-trivial FD.
  4. Both $A \rightarrow B$ and $B \rightarrow A$ hold in $R$. 
Decomposition into BCNF

- Suppose \( R \) is a relation schema that violates BCNF.
- We can decompose \( R \) into a set \( S \) of new relations such that
  1. each relation in \( S \) is in BCNF and
  2. we can “recover” \( R \) from the relations in \( S \), i.e., the relations in \( S \) “faithfully” represent the data in \( R \).
- How does the normalisation algorithm guarantee the second condition?
Candidate Normalization Algorithm

- Every two-attribute relation is in BCNF.
- Can we bring any relation $R$ into BCNF by arbitrarily decomposing it into two-attribute relations?
- No, since we may not be able to recover $R$ correctly from the decomposition.
Joining Relations

Let $R$ and $S$ be two relations with one common attribute $B$.

Relation $T$ is the join of $R$ and $S$, denoted $R \bowtie S$ if and only if

- the attributes of $T$ are the union of the attributes of $R$ and $S$,
- every tuple $t \in T$ is the join of two tuples $r \in R$ and $s \in S$ that agree on the attribute $B$, i.e., $t$ agrees with $r$ on all the attributes in $R$ and with $s$ on all attributes in $S$,
- $T$ contains all tuples formed in this manner.
Recovering Information from a Decomposition

- Suppose \( R \) is a relation schema that violates BCNF.
- We can decompose \( R \) into a set \( \{S_1, S_2, \ldots S_k\} \) of new relations such that
  
  1. each relation \( S_i, 1 \leq i \leq k \) is in BCNF and
  2. we can “recover” \( R \) from these relations:
     
     \[ R = S_1 \bowtie S_2 \bowtie \ldots \bowtie S_k, \text{ i.e., the decomposition of } R \text{ into} \]
     
     \( \{S_1, S_2, \ldots S_k\} \) is a \textit{lossless-join} decomposition.
Correct Decompositions

A decomposition is *lossless* if we can recover:

$$R(A, B, C)$$

Decompose

$$R_1(A, B)$$  $$R_2(A, C)$$

Recover

$$R'(A, B, C)$$ should be the same as $$R(A, B, C)$$

$$R'$$ is in general larger than $$R$$. Must ensure $$R' = R$$
Example of Lossy-Join Decomposition

- Example: Decomposition of $R = (A, B)$
  $$R_1 = (A) \quad R_2 = (B)$$

\[
\begin{array}{c|c}
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\end{array}
\quad
\begin{array}{c}
A \\
\hline
\alpha \\
\beta \\
\end{array}
\quad
\begin{array}{c}
B \\
\hline
1 \\
2 \\
\end{array}
\]

$\Pi_A(r) \Join \Pi_B(r)$
Example: BCNF Decomposition

Drinkers(name, addr, beersLiked, manf, favBeer)

FDs = name->addr, name -> favBeer, beersLiked->manf

• Pick BCNF violation name->addr.
• Close the left side: \{name\}^+ = \{name, addr, favBeer\}.
• Decomposed relations:
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers2(name, beersLiked, manf)
Example -- Continued

• We are not done; we need to check Drinkers1 and Drinkers2 for BCNF.

• Is Drinkers1 in BCNF?
  – For Drinkers1(\texttt{name, addr, favBeer}), relevant FD’s are \texttt{name->addr} and \texttt{name->favBeer}.
  – Thus, \{\texttt{name}\} is the only key and Drinkers1 is in BCNF.
Example -- Continued

• For Drinkers2(name, beersLiked, manf), the only FD is beersLiked->manf, and the only key is 
  \{name, beersLiked\}.
  – Violation of BCNF?
• beersLiked⁺ = \{beersLiked, manf\}, so we decompose Drinkers2 into:
  1. Drinkers3(beersLiked, manf)
  2. Drinkers4(name, beersLiked)
Example -- Concluded

• The resulting decomposition of *Drinkers*:
  1. Drinkers1(name, addr, favBeer)
  2. Drinkers3(beersLiked, manf)
  3. Drinkers4(name, beersLiked)

• Note:
  – *Drinkers1* tells us about drinkers,
  – *Drinkers3* tells us about beers, and
  – *Drinkers4* tells us the relationship between drinkers and the beers they like.
Summary of BCNF Decomposition

Find a dependency that violates the BCNF condition:

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

Decompose:

- **Others**
- **A's**
- **B's**

Is there a 2-attribute relation that is not in BCNF?

Continue until there are no BCNF violations left.