CSI T5300: Advanced Database Systems

L06: Relational Database Design – BCNF

Dr. Kenneth LEUNG

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China
• **1NF**: all attribute values are atomic (every relational table is in 1NF)
• **2NF**: for every FD $X \rightarrow A$, either $X$ is not a proper subset of a candidate key, or every attribute in $A$ is prime
• **3NF**: for every FD $X \rightarrow A$, either $X$ is a superkey, or every attribute in $A$ is prime
  - Difference between 3NF and 2NF: 2NF allows FDs where $X$ is not part of any candidate key
Let $R$ be a relation schema, with the set $F$ of FDs. $R$ is in **BCNF** if and only if

- for each FD: $X \rightarrow A$ in $F^+$
  - Then
    - $A \subseteq X$ (the FD is trivial), or
    - $X$ is a superkey for $R$

**In words:** For every FD that does not contain extraneous (useless) attributes, the LHS of every FD is a candidate key.

- BCNF tables have **no redundancy**
- If a table is in BCNF, it is also in 3NF (and 2NF and 1NF)
• $R = (B, C, E)$
  \[ F = \{\{E\}\rightarrow\{B\}, \{B,C\}\rightarrow\{E\}\}\]

• $R$ is in 3NF

• **Two candidate keys:** BC and EC
  \[\{B,C\}\rightarrow\{E\}\] does not violate BCNF because BC is a key
  \[\{E\}\rightarrow\{B\}\] violates BCNF because E is not a key

• In order to achieve BCNF we have to decompose the table, but how?
  - Since the decomposition must be **lossless**, we only have one option: $R1(B,E)$, and $R2(C,E)$. The common attribute E should be key of one fragment, here $R1$. 
• Bank-schema = (Branch B, Customer C, Employee E)
• F = \{\{E\}\rightarrow\{B\}, \{B,C\}\rightarrow\{E\}\}
• Decompose into R1(B,E), and R2(C,E)

<table>
<thead>
<tr>
<th>Branch</th>
<th>Customer</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKUST</td>
<td>Wong</td>
<td>Au</td>
</tr>
<tr>
<td>HKUST</td>
<td>Chin</td>
<td>Au</td>
</tr>
<tr>
<td>Central</td>
<td>Wong</td>
<td>Jones</td>
</tr>
<tr>
<td>Central</td>
<td>null</td>
<td>Cheng</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wong</td>
<td>Au</td>
</tr>
<tr>
<td>Chin</td>
<td>Au</td>
</tr>
<tr>
<td>Wong</td>
<td>Jones</td>
</tr>
</tbody>
</table>
• We can generate the original table by joining the two fragments, using an **outer join**

![Table: Branch, Employee](image)

<table>
<thead>
<tr>
<th>Branch</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKUST</td>
<td>Au</td>
</tr>
<tr>
<td>Central</td>
<td>Jones</td>
</tr>
<tr>
<td>Central</td>
<td>Cheng</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wong</td>
<td>Au</td>
</tr>
<tr>
<td>Chin</td>
<td>Au</td>
</tr>
<tr>
<td>Wong</td>
<td>Jones</td>
</tr>
</tbody>
</table>

**=**

<table>
<thead>
<tr>
<th>Branch</th>
<th>Customer</th>
<th>Employee</th>
</tr>
</thead>
<tbody>
<tr>
<td>HKUST</td>
<td>Wong</td>
<td>Au</td>
</tr>
<tr>
<td>HKUST</td>
<td>Chin</td>
<td>Au</td>
</tr>
<tr>
<td>Central</td>
<td>Wong</td>
<td>Jones</td>
</tr>
<tr>
<td>Central</td>
<td>null</td>
<td>Cheng</td>
</tr>
</tbody>
</table>

• Is the decomposition dependency preserving?
  - No. We lose \{B,C\}→\{E\}

• Can we have a dependency preserving decomposition?
  - No. No matter how we break, we lose \{B,C\}→\{E\} since it involves all attributes
Observations about BCNF

• It avoids the problems of redundancy and all anomalies
• There is always a lossless decomposition that generates BCNF tables
• However, we may not be able to preserve all dependencies

• Next step: an algorithm for automatically generating BCNF tables
Algorithm for BCNF Decomposition

- **Input:** The initial table \( R \) with FDs \( F \)
- **Output:** The table decomposition \( S \) into BCNF

\[
S = \{ R \}
\]

**while** not all relation schemes in \( S \) are in BCNF

find \( R \) in \( S \) with \( FD \ X \rightarrow Y \) that violates BCNF for \( R \)

\[
S = (S - \{ R \}) \cup \{ R-Y \} \cup \{(X,Y)\}
\]

- This is a simplified version
- **In words:**
  - When we find a table \( R \) with BCNF violation \( X \rightarrow Y \) we:
    1] Remove \( R \) from \( S \)
    2] Add a table that has the same attributes as \( R \) except for \( Y \)
    3] Add a second table that contains the attributes in \( X \) and \( Y \)
Relation schema: \( R = (A, B, C, D, E) \)
- FDs: \( \{A\} \rightarrow \{B, E\}, \{C\} \rightarrow \{D\} \)
- Candidate key: AC

Both functional dependencies violate BCNF because the LHS is not a candidate key
- Pick \( \{A\} \rightarrow \{B, E\} \)
  - We can also choose \( \{C\} \rightarrow \{D\} \). Different choices lead to different decompositions.
- \( R = (A, B, C, D, E) \) generates \( R_1 = (A, C, D) \) and \( R_2 = (A, B, E) \)

Do we need to decompose further?
R1=(A,C,D) and R2=(A,B,E)

{A}→{B,E}, {C}→{D}

We need to decompose R1=(A,C,D) because of the FD {C}→{D}

Thus (A,C,D) is replaced with R3=(A,C) and R4=(C,D).

Final decomposition: S = {R2=(A,B,E), R3=(A,C), R4=(C,D)}

Is the decomposition lossless?
- Yes the algorithm always creates lossless decompositions. In step $S = (S - \{R\}) \cup (R-Y) \cup (X,Y)$ we replace R with tables (R-Y) and (X,Y) that have X as the common attribute and $X \rightarrow Y$, i.e., X is the key of (X,Y)

Is the decomposition dependency preserving?
- Yes, because $F2=\{A\rightarrow B,E\}$, $F3=\emptyset$, $F4=\{C\rightarrow D\}$ and $(F2 \cup F3 \cup F4)^+ = F^+$

But remember: sometimes we may not be able to preserve dependencies!
**Important question:**
- Which dependencies to check for BCNF violations? $F$ or $F^+$?

**Answer-Part 1:**
- To check if a table $R$ with a given set of FDs $F$ is in BCNF, it suffices to check only the dependencies in $F$.
- Consider $R(A, B, C, D)$, with $F = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}\}$
  - The key is $\{A, D\}$
  - $R$ violates BCNF because the LHS of both $\{A\} \rightarrow \{B\}$ and $\{B\} \rightarrow \{C\}$. Neither $A$ nor $B$ is a key.
  - We can see that by simply using $F$ - we do not need $F^+$ (e.g., we do not need to check the implicit FD $\{A\} \rightarrow \{C\}$).
- We can show that if none of the dependencies in $F$ causes a violation of BCNF, then none of the dependencies in $F^+$ will cause a violation of BCNF either.
Answer-Part 2:
- However, using only $F$ is insufficient when testing a fragment in the decomposition of $R$
- Consider again $R(A,B,C,D)$, with $F = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}\}$ that violates BCNF
  - Decompose $R$ into $R_1(A,C,D)$ and $R_2(A,B)$
  - There is no FD in $F$ that contains only attributes from $R_1(A,C,D)$ so we might be misled into thinking that $R_1$ is in BCNF.
  - In fact, dependency $\{A\} \rightarrow \{C\}$ in $F^+$ shows that $R_1$ is not in BCNF.
  - Therefore, for the decomposed relations we also need to consider dependencies in $F^+$ (see next slide).
To check if a relation $R_i$ in a decomposition of $R$ is in BCNF,

- Either test $R_i$ for BCNF with respect to the restriction of $F^+$ to $R_i$ (that is, all FDs in $F^+$ that contain only attributes from $R_i$)
- or use the following test:
  - for every set of attributes $X \subseteq R_i$, check that the attribute closure $X^+$ either includes no attribute of $R_i - X$, or includes all attributes of $R_i$.
  - If the condition is violated, the dependency $X \rightarrow (X^+ - X) \cap R_i$ holds on $R_i$, and $R_i$ violates BCNF.
  - We use the above dependency to decompose $R_i$

  - Note: we have seen how to compute $X^+$ in the slide set about FDs
• Consider again: R(A,B,C,D), F = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}\} and the decomposition R1(A,C,D) and R2(A,B)

• A+={A,B,C}, B+={B,C}, C+={C}
  – R2(A,B) is in BCNF because
    • A+ \cap R2 = \{A,B,C\} \cap \{A,B\} = \{A,B\} includes all attributes of R2
    • B+ \cap R2 = \{B,C\} \cap \{A,B\} = \{B\} includes no attributes of R2-\{B\}
    • In other words, each attribute (e.g., A) determines everything (it is a key) or nothing (e.g., B).
  – R1(A,C,D) is not in BCNF because
    • A+ \cap R1 = \{A,B,C\} \cap \{A,C,D\} = \{A,C\} does not include all attributes of R1
    • Therefore, the dependency \{A\} \rightarrow \{C\} causes a BCNF violation and will be used for further decomposing R1
    • Final decomposition: R2(A,B), R3(A,D), R4(A,C)
Different BCNF Decompositions

- The different possible orders in which we consider FDs violating BCNF in the algorithm may lead to different decompositions.

- Previous example: $R(A,B,C,D)$, $F = \{\{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}\}$
- Previous BCNF decomposition: $R2(A,B), R3(A,D), R4(A,C)$

- Question: is the decomposition dependency preserving?
- Answer: No – we lost the dependency $\{B\} \rightarrow \{C\}$
- Question: Can you obtain a dependency preserving decomposition?
- Answer: Yes – in the first decomposition, we first handled violation $\{A\} \rightarrow \{B\}$. If, instead, we apply $\{B\} \rightarrow \{C\}$ we obtain:
  - $R1=(A,B,D)$ and $R2=(B,C)$
  - We decompose $R1=(A,B,D)$ further using $\{A\} \rightarrow \{B\}$ to obtain:
    - $R3=(A,D)$ and $R4=(A,B)$
  - The final decomposition $R2=(B,C), R3=(A,D), R4=(A,B)$ is dependency preserving.
Normalization Goals

• Goal for a relational database design:
  - BCNF (i.e., no redundancy)
  - Lossless join
  - Dependency preservation

• If we cannot achieve this, we accept one of
  - Lack of dependency preservation in BCNF
  - Redundancy due to the use of 3NF
• When an E-R diagram is carefully designed, the tables generated from the E-R diagram should not need further normalization.

• However, in a real (imperfect) design, there can be FDs from non-key attributes of an entity to other attributes of the entity.

• Example:
  - Employee entity with attributes department-number and department-address, and an FD department-number → department-address.
  - A Good design would have made department an entity.
• We start with a single universal relation and we decompose it using the FDs (no ER diagrams)

• Assume Loans(branch-name, loan-number, amount, customer-id, customer-name) and FDs:
  - \{loan-number\} → \{branch-name, amount, customer-id\}
  - \{customer-id\} → \{customer-name\}

• We apply existing decomposition algorithms to generate tables:
  - Loan(loan-number, branch-name, amount, customer-id)
  - Customer(customer-id, customer-name)
We may want to use non-normalized schema for performance. E.g., displaying customer-name along with loan-number and amount requires join of Loan with Customer.

Alternative 1: Use denormalized relation containing attributes of Loan as well as Customer with all above attributes
- faster lookup
- extra space and extra execution time for updates
- extra coding work for programmer and possibility of error in extra code

Alternative 2: use a materialized view defined as Loan JOIN Customer
- Benefits and drawbacks same as above, except no extra coding work for programmer and avoids possible errors
• Some aspects of database design are not caught by normalization

• E.g., instead of \texttt{earnings(company-id, year, amount)}, use
  \begin{itemize}
    \item \texttt{earnings-2000, earnings-2001, earnings-2002}, etc., all on the schema \texttt{(company-id, earnings)}.
      \begin{itemize}
        \item Above are in BCNF, but make querying across years difficult and needs new table each year
      \end{itemize}
    \item \texttt{company-year(company-id, earnings-2000, earnings-2001, earnings-2002)}
      \begin{itemize}
        \item Also in BCNF, but also makes querying across years difficult and requires new attribute each year.
      \end{itemize}
  \end{itemize}